目　录

Webhook Data Push Service

Webhook Client code example

- .NET（C#）
- Java
- Webhook Server-Python Code

MQTT Data Push Service

MQTT Client code example

- .NET(C#)
- Java
- Python

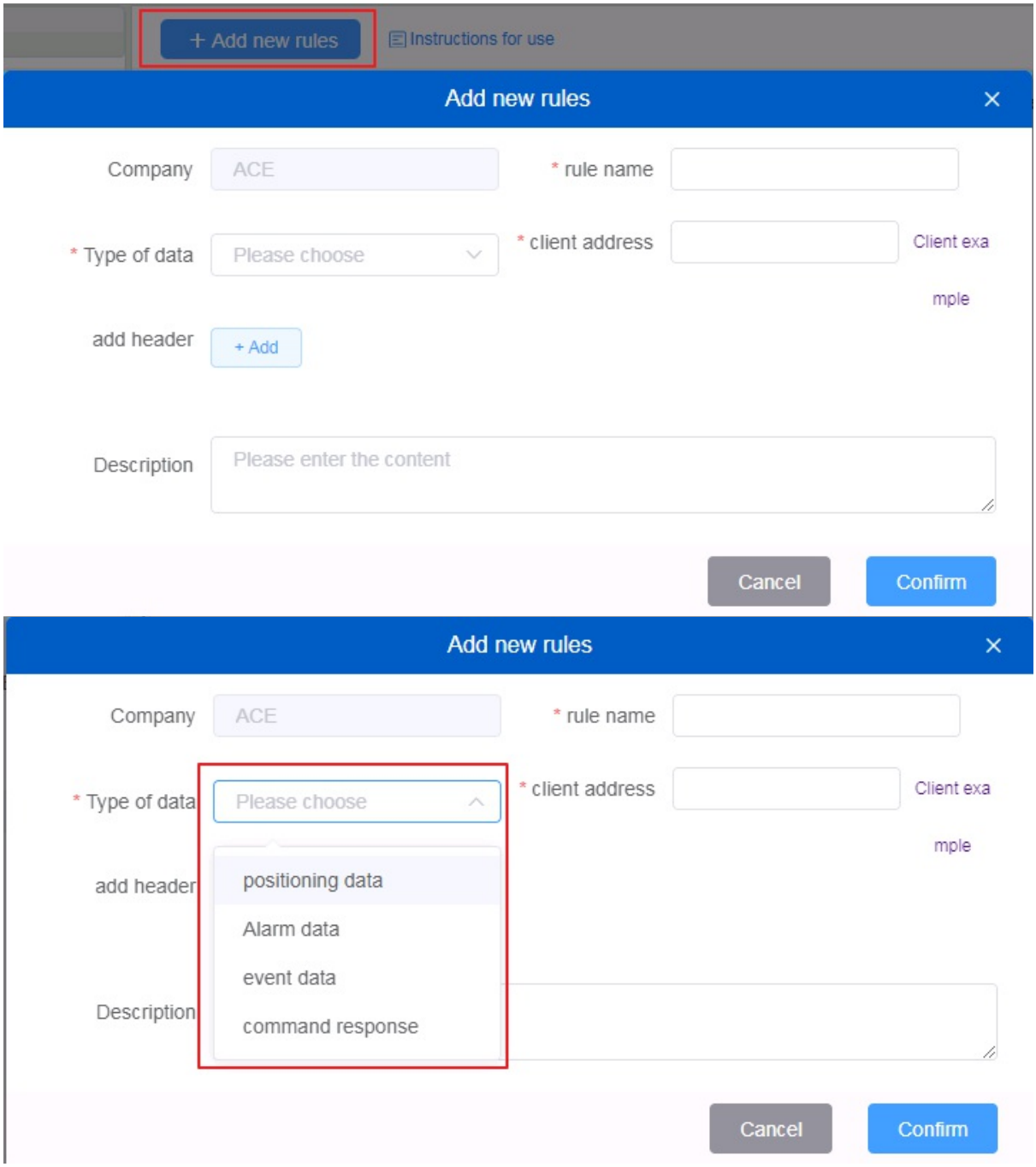
Webhook Data Push Service

Describe

```
The data push service is mainly used to meet the needs of customers for sec
ondary development of basic devices data. This function can realize the act
ive push of devices location data, alarm data, event data, offline/online s
tatus data, and data issued by commands.
1. First, you need to write a WEB API interface based on the POST method to
 receive the data forwarded by the server. Please refer to the client code
example
2. Configure the written WEBAPI address through the rule ->Create a new rul
e, and fill it into the client address, which corresponds to the full URL o
f the interface, such as: http://47.112.122.222:8080/revTerminalLocation
3. After the rules are configured, they are expected to take effect automat
ically within 5 minutes. At this time, the device reports the data, and the
 system's push engine will forward the data to the configured client addres
s.
```

Rule Configuration

The user can configure the data to be pushed by the specified company in the rule configuration interface. The user needs to configure the data type, client address and other information.
If there is a message header restriction, you need to add the message header configuration. If not, you can ignore it.



Data type specification

1.Positioning data

```
Device positioning data, including Device ID, positioning time, longitude a
nd latitude, speed, mileage and other information
```

JSON String Example

```
 {"assetId":"8022300001","longitude":0.0,"latitude":0.0,"speed":0,"directi
```

```
on":0,"mileage":16,"gpsTime":"2022-09-01T08:59:56Z",\"ksInfo\":\"null\","re
cvTime":"2022-09-01T08:59:56.964Z","locType":0,"cellSignal":19,"gnssSignal"
:0,"cells":"460,0,9383,220792647","battery":100,"voltage":"0.0","statusJson
":"{\"lockBar\":-1,\"lockRope\":0,\"lockStatus\":0}","expandInfo":"{\"angle
\":\"null\",\"backBattery\":\"null\",\"fuels\":\"-1,-1,-1\",\"humidity\":\"
0\",\"lux\":\"0.0\",\"networkType\":\"0\",\"pressure\":\"0.0\",\"reportType
\":\"0\",\"temperature\":\"-1000.0\"}"}
```


Json Field Description

Parameter name	Type	Description
assetId	String	Device ID
longitude	Double	longitude(WGS-84)
latitude	Double	latitude(WGS-84)
speed	Integer	speed(km/h)
direction	Integer	direction(0~360)
mileage	Long	mileage(km)
gpsTime	String	Positioning time（UTC）
recvTime	String	Receiving time（UTC）
locType	Integer	Positioning type (0: no positioning; 1: GPS positioning; 2: base statio
cellSignal	Integer	GPRS signal
gnssSignal	Integer	Satellite signal
cells	String	Cell code data is displayed in MNC, MCC, LAC and CID formats
battery	Integer	Power, 255 indicates charging
voltage	String	Voltage(V)
statusJson	String	Lock/vehicle mounted status JSON (refer to the status JSON descript
expandInfo	String	Extension information JSON (refer to the following extension inform
		KS20A asset information JSON（refer to the following KS20A asset i description）,eg:,"ksInfo":"{"angVelocity":21000.0,"eleStatus":0,"eleV

		{"112408000035":{"dateTime":"2025-05-14T09:39:03Z","humidity":60.0,"lux":0.0,"subId":"112408000035","tem

Lock Status JSON Description

Parameter name	Type	Description
lockRope	Integer	Rope locking status (1: pulling out; 0: inserting; - 1: none)
lockStatus	Integer	Lock status (0: Off; 1: On)

Vehicle status JSON description

Parameter name	Type	Description
acc	Integer	Engine switch status (1: On; 0: Off; 1: None)
fuelCut	Integer	State of oil cut-off electric switch (1: On; 0: Off; 1: None)
door	Integer	Door opening/closing status (1: open; 0: close; - 1: none)
engine	Integer	Engine status (1: On; 0: Off; 1: None)

JSON description of extension information

Parameter name	Type	Description
temperature	String	Temperature - 1000 means none, unit: °C
humidity	String	Humidity, unit :%
fuels	String	Oil level value - 1 means none (“- 1, - 1, - 1”)
fAcceleration	String	acceleration(formats:”x: 1; y: - 29; z: - 2903”)
lux	float	Illuminance(lux)
pressure	float	pressure(pa)
posture	String	Posture (formats:”x: 1; y: - 29; z: - 2903”)
fVoltage	Double	Voltage value(V)
backBattery	String	Backup battery (“55,3.88,0”)

fReportType	Integer	Data type (0: real-time; 1: supplementary report; 2: alarm)
fNetworkType	Integer	Network type (0: unknown 1:1G 2:2G 3:3G 4:4G 5:5G)

JSON description of KS20A asset information

Parameter name	Type	Description
temperature	double	Onboard temperature -1000 means none, unit: °C
humidity	Integer	humidity, unit :%
volStatus	Integer	Voltage status (0: Normal LED voltage 1: Overvoltage LED voltage 2: Undervoltage voltage)
volValue	double	Voltage value, unit :V
eleStatus	Integer	Current status 0: LED current normal 1: LED current overcurrent 2: Current undercurrent
eleValue	double	Current value, unit :A
pressure	float	Air pressure (pa)
shakeStatus	Integer	0: Stationary 1: Shaking
angVelocity	Long	Angular velocity, unit :mdps
posture	String	Posture x,y,z axis values
vibrate	Integer	Amplitude, unit :mg
subLockId	String	Lock controller ID
subLargeCharId	String	Large word controller ID
luminance1	Integer	Luminance of white light, unit :%
luminance2	Integer	Red light brightness, unit :%
luminance3	Integer	Yellow light luminance, unit :%
luminance4	Integer	Blue luminance, unit :%
subInfo	Map < String,	peripherals information, parameter values: {\ “112408000042 " : {\ “dateTime " : \ “the 2025-05-13 T09:55:25 Z ", \

subInfo	Object >	“eventType " : 1, \ “humidity " : 54. 0, \ “lux " : 0.0, \ “subId " : \ “112408000042 ", \ “temperature " : 28.9}}

description of subInfo

Parameter name	Type	Description
subId	String	Peripheral ID
dateTime	String	Peripheral online time,UTC time
temperature	double	Temperature, unit :°C
humidity	Integer	humidity, unit :%
lux	double	Illuminance, unit :lux

2.Alarm data

```
Alarm data generated by equipment, including equipment number, alarm type,
alarm time, longitude and latitude, etc.
```

Push JSON Example

```
{"assetId":"875011000025","alarmType":46,"dateTime":"2020-04-01T00:00:13Z",
"longitude":0.0,"latitude":0.0,"speed":0,"mileage":0,"cells":"0,0,0,0","des
cribe":"","fileIndex":"1585699212"}
```

Json Field Description

Parameter name	Type	Description
assetId	String	Device ID
alarmType	Integer	Alarm type (refer to the following description of alarm type)
dateTime	String	Time (UTC time)
longitude	Double	longitude(WGS-84)
latitude	Double	latitude(WGS-84)

mileage	Long	mileage(km)
cells	String	Cell code data is displayed in MNC, MCC, LAC and CID formats
describe	String	alarm describe
fileIndex	String	Serial number of files

Alarm Type Description：

Alarm Type	Alarm Name
1	Speeding Alarm
2	Fatigue driving
3	Danger warning
4	GNSS module failure
5	GNSS antenna not connected or cut off
6	GNSS antenna short circuit
7	device main power supply under voltage
8	Power failure of device main power supply
9	device LCD or display failure
10	TTS module failure
11	Camera fault
12	IC card module failure
13	Speeding warning
15	Power failure
19	Overtime parking
23	Route deviation alarm
24	Vehicle VSS failure

24	Vehicle VSS failure
26	Vehicle theft
27	Illegal vehicle ignition
28	Illegal vehicle displacement
29	Collision warning
30	Rollover warning
31	Illegal door opening alarm
32	Video signal loss alarm
33	Video signal blocking alarm
34	Storage unit failure alarm
35	Other video equipment fault alarm
36	Passenger train overload alarm
37	Abnormal driving behavior alarm
38	Video recording reaches storage threshold alarm
40	Lock rope cut
41	Vibration
42	Long-time unlocking
43	Unlock password error for 5 consecutive times
44	Swiping illegal RFID card
45	Low power
46	Back cover opened
47	Motor stuck
48	Enter fence alarm
49	Exit fence alarm

51	SOS
52	Towing alarm
54	Fuel level change alarm
55	Enter hot spot alarm
56	Exit hot spot alarm
57	Access road alarm
58	Exit road alarm
63	Temperature and humidity alarm
64	Forward collision alarm
65	Lane departure alarm
66	Too close distance alarm
67	Pedestrian collision alarm
68	Frequent lane change alarm
69	Road sign overrun alarm
70	Obstacle alarm
71	Road sign recognition events
72	Active capture event
73	Lane changing of solid line
74	Pedestrian monitoring in car aisle
92	Fatigue driving alarm
93	Call the police
94	Smoking alarm
95	Distracted driving alarm
96	Driver abnormal alarm

96	Driver abnormal alarm
97	Auto capture events
98	Driver change event
99	Probe blocked
100	Overtime driving
101	Not wearing seat belt
102	Infrared blocking sunglasses failure
103	Hands off the steering wheel
104	Playing with phone
110	Tire pressure alarm
125	Rear approach alarm
126	Left rear approach alarm
127	Right rear approach alarm
131	Rapid acceleration
132	Emergency deceleration
133	Emergency turning
134	Idle alarm
135	Abnormal flameout
136	Neutral skid alarm
137	Engine overspeed
141	Overspeed
145	Excess load of vehicle
146	Excess load of road
151	Height limit alarm

161	Emergency brake alarm
162	Neutral skating
163	GPRS reconnection
164	Dispatching panel connection
165	Dispatch panel disconnected
166	CANbus disconnection alarm
167	CANbus fault code upload alarm
168	Limit drive alarm
180	Host dismantling
190	Damage of upper cover
191	GPS antenna interference
192	Low power sleep alarm
193	Device exception
194	Unlock password error
195	Do not perform unlocking without positioning
196	No unlocking outside the fence
197	Attitude alarm
198	Slave sensor signal lost
199	Supervision status forbids unlocking
200	Main battery replacement
201	Self-check exception
202	not enough storage space
203	Lock bar bent
204	Chain saw off

204	Chain saw off
205	lock bar missing
206	Forbidden to unlock in the sealed state
316	Light alarm
332	Pressure alarm
400	Idling alarm
402	Fence stop alarm
403	Out fence unlock
404	Departure delay warning
405	Departure delay alarm
406	Late warning
407	Late alarm

3.Event data

```
Event data generated by Device, including Device ID, event type, time, long
itude and latitude, etc.
```

Push JSON Example


```
{"assetId":"795206001104","eventType":1,"unLockType":1,"dateTime":"2022-08-
19T02:07:28Z","longitude":0.0,"latitude":0.0,"speed":0,"mileage":5,"cells":
"460,0,10352,220726855","card":"0003116836","password":"","describe":"{"way
billGUID":"190db1a2-3df1-4857-b94e-ee5937334409","waybillNo":"45435345"}}
```


Json Field Description

Parameter name	Type	Description
assetId	String	Device ID
eventType	Integer	Event type (refer to the following description of event type)

		unlock 5: SMS unlock 6: regional trigger)
dateTime	String	Time (UTC time)
longitude	Double	longitude(WGS-84)
latitude	Double	latitude(WGS-84)
speed	Integer	speed(km/h)
mileage	Long	mileage(km)
cells	String	Cell code data is displayed in MNC, MCC, LAC and CID formats
card	String	Swipe card No,only for lock products
password	String	unlock password,only for lock products
describe	String	Description information. For route analysising waybill events 13-18, the description will have a Json string value, showing the waybill unique identifier (waybillGUID) and waybill number(Declaration No.)

Event Type Description：

Event Type	Event Name
1	Unlock
2	Lock
3	Sealing of exit area
4	Unblock the entrance area
5	Open box(door)
6	Close box(door)
7	Bluetooth sealing
8	Bluetooth unblocking
9	Remote sealing
10	Remote unsealing

10	Remote unsealing
11	Snap photos
12	Timed shooting
13	leave the starting point
14	reach the destination
15	leave the destination
16	Complete the waybill
17	entry waypoint
18	leave waypoint
19	Pull out the locking rope / Press the unlocking button(JT705A/JT705C)

4.Command response data

```
Command response data generated by the device, including device number, com
mand type, time, and returned content information of the command.
```

Push JSON Example

```
{"assetId":"742207000010","commandType":"BASE2","content":"[\"742207000010\
",\"8\",\"001\",\"BASE\",\"2\",\"20220901081402\"]","dateTime":"2022-09-01T
08:14:03.564Z"}
```

Json Field Description

Parameter name	Type	Description
assetId	String	Device ID
commandType	String	Command type
content	String	Equipment response content
dateTime	String	Time (UTC time)

5.Slave lock(sensor)

```
itude and latitude, speed, mileage and other information
```

JSON String Description

```
 {"assetId":"8012600005","longitude":113.92294,"latitude":22.670017,"speed
":0,"direction":0,"gpsTime":"2023-12-07T08:30:31Z","recvTime":"2023-12-07T0
8:30:52.474Z","subGpsTime":"2023-12-07T08:29:50Z","battery":71,"voltage":"3
.93","subAssetID":"E0171E0365","sensorType":4,"statusJson":"{\"lockRope\":0
,\"gateway\":0}","locStatus":0,"locRope":0,"rssi":57,"temperature":-1000.0,
"humidity":0,"eventType":6,"locTimes":5,"subGpsTimestamp":1701937790000}
```

JsonField Description

Parameter name	Type	Description
assetId	String	Device ID
longitude	Double	longitude(WGS-84)
latitude	Double	latitude(WGS-84)
speed	Integer	speed(km/h)
direction	Integer	direction(0~360)
mileage	Long	mileage(km)
gpsTime	String	Positioning time（UTC）
recvTime	String	Receiving time（UTC）
subAssetID	String	Slave Equipment No
subGpsTime	String	Slave Positioning time（UTC））
subGpsTimestamp	Long	Slave Positioning timestamp
battery	Integer	Slave Power
voltage	String	Slave voltage (V)
sensorType	Integer	Slave Type 1-JT126 , 4-JT709 , 5-JT801 , 6-JT802
locStatus	Integer	Slave lock status 0-Off 1-On
locRope	Integer	Slave Rope locking status (1: pulling out 0: inserting - 1: none)

locRope	Integer	Slave Rope locking status (1: pulling out 0: inserting - 1: none)
rssi	Integer	RSSI
humidity	Integer	Slave Temperature - 1000 means none
temperature	double	Slave Humidity 0 means none
locTimes	Integer	Slave unlock times
eventType	Integer	Slave Event -1：none 0:Lock event 1:BLE unlock event 2:Open back cover waring 3:Lora unlock event 4:Lock rope cut warning 5:Key wake up event 6:Heartbeat packet 7：Charging report event 8/20:Pull out the lock rope event 9：RFID unlock event 10：Illegal RFID unlock warning 14：Sub lock signal lost warning 15：Valve closed event 16：Valve open event 17：Low battery warning 18：Anti-disassembly 19：Disassembly of electronic compartment 21：Lock rope insertion 22：Bluetooth connection 23：Open emergency storage warning 24：Close emergency storage warning 25：Valve abnormally open warning 26：Turn off the knob event 27：Turn on the knob event 28:Unlocking Error 29:Abnormal Motor 30:NFC trigger
statusJson	String	For 802 device status data (Json string), other types of device data can be ignored

statusJson Field Description

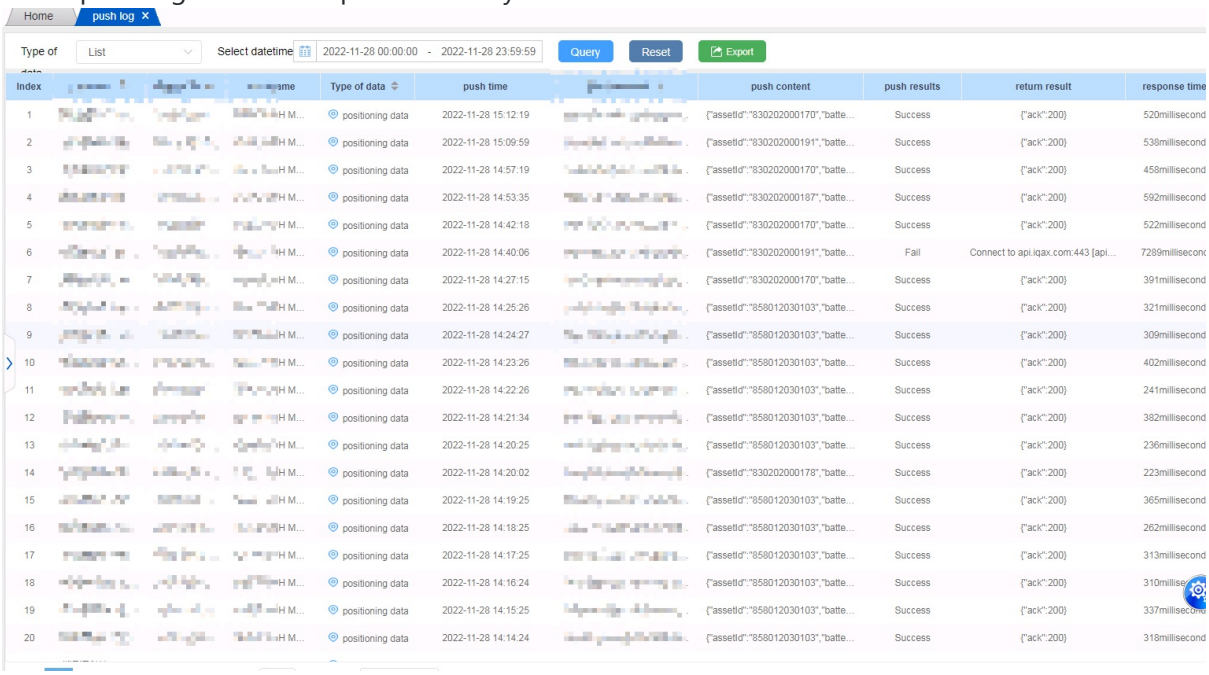
“FStatusJson”:”{"bottomDisassembly":0,"emergencyKey":0,"lockMotor":0,"structuralDisassembly":0,"gateway":0,"lockValve":0}”

Parameter name	Type	Description
bottomDisassembly	Integer	bottom disassembly status,0:No disassembly，1:disassembly
emergencyKey	Integer	emergency key status 0：Seal up，1：Enable
lockMotor	lock motor status：0：Off，1：On	
		structural disassembly status,0:No

lockKnob	Integer	Lock Knob status，0：Off，1：On
lockRope	Integer	Rope locking status 0：inserting；1：pulling out
gateway	Integer	Meaningless, can be ignored

Push log report

After configuring the push rules, users can query the logs generated by the platform's active push in the push log to facilitate problem analysis.



Webhook Client code example

.NET(C#)

Java

Python

.NET (C#)

Example Description

```
Some field data may be meaningless due to different equipment types. If it
is not available, it can be ignored
```

1.Positioning data

```
[HttpPost]
public MBackResult ReceiveLocationData([FromBody]TerminalLocation parm)
{
    Log.Instance.Info("Positioning data：" + JsonConvert.SerializeObject(pa
rm));
    MBackResult result = new MBackResult();
    result.Result = 200;
    return result;
}
```

TerminalLocation:

```
    /// <summary>
    /// Positioning data
    /// </summary>
    public class TerminalLocation
    {
        /// <summary>
        /// Equipment No
        /// </summary>
        public string assetId { get; set; }
        /// <summary>
        /// longitude(WGS-84)
        /// </summary>
        public double longitude { get; set; } = 0.0;
        /// <summary>
        /// latitude(WGS-84)
        /// </summary>
        public double latitude { get; set; } = 0.0;
        /// <summary>
        /// speed(km/h)
        /// </summary>
```

```
        public int speed{ get; set; }
        /// <summary>
        /// direction(0~360)
        /// </summary>
        public int direction { get; set; } = 0;
        /// <summary>
        /// mileage(km)
        /// </summary>
        public long mileage { get; set; } = 0L;
        /// <summary>
        /// Positioning time ( UTC )
        /// </summary>
        public string gpsTime { get; set; }
        /// <summary>
        /// Receiving time ( UTC )
        /// </summary>
        public string recvTime { get; set; }
        /// <summary>
        /// Positioning type (0: no positioning; 1: GPS positioning; 2: bas
 e station positioning;3:Cells positioning)
        /// </summary>
        public int locType { get; set; } = 0;
        /// <summary>
        /// GPRS signal
        /// </summary>
        public int cellSignal { get; set; } = 0;
        /// <summary>
        /// Satellite signal
        /// </summary>
        public int gnssSignal { get; set; } = 0;
        /// <summary>
        /// Cell code data, MNC, MCC, LAC, CID
        /// </summary>
        public string cells { get; set; }
        /// <summary>
        /// Power
        /// </summary>
        public int battery { get; set; }
        /// <summary>
        /// Voltage(V)
        /// </summary>
        public string voltage { get; set; }
        /**
         * Lock/vehicle status JSON
         */
        public string statusJson { get; set; }
```

```
        /**
         * Extension information JSON
         */
        public string expandInfo { get; set; }
    }
```

Lock/vehicle status JSON

```
    /// <summary>
    /// Lock/vehicle status
    /// </summary>
    public class StatusJson
    {
      /// JSON description of lock status information (for electronic lock
devices)
        /// <summary>
        /// Rope locking status (1: pulling out 0: inserting - 1: none)
        /// </summary>
        public int lockRope { get; set; }
        /// <summary>
        /// Lock status (0: Off 1: On)
        /// </summary>
        public int lockStatus { get; set; }


    ///JSON description of on-board status information (for GP series, mini
sterial equipment and other vehicle equipment)
        /// <summary>
        /// ACC status (1: On 0: Off 1: None)
        /// </summary>
        public int acc { get; set; }
        /// <summary>
        /// State of oil cut-off electric switch (1: On 0: Off 1: None)
        /// </summary>
        public int fuelCut { get; set; }
        /// <summary>
        /// Door opening/closing status (1: open 0: close - 1: none)
        /// </summary>
        public int door { get; set; }
        /// <summary>
        /// Engine status (1: On 0: Off 1: None)
        /// </summary>
        public int engine { get; set; }
        }
```

JSON description of extension information

```
    /// <summary>
    /// Extended Fields
    /// </summary>
    public class ExpandInfo
    {
        /// <summary>
        /// Temperature - 1000 means none
        /// </summary>
        public int temperature { get; set; }
        /// <summary>
        /// Humidity 0 means none
        /// </summary>
        public int humidity { get; set; }
        /// <summary>
        /// Oil level value - 1 means none ("- 1, - 1, - 1")
        /// </summary>
        public int fuels { get; set; }
        /// <summary>
        /// acceleration(formats:"x: 1; y: - 29; z: - 2903")
        /// </summary>
        public string acceleration { get; set; }
        /// <summary>
        /// Illuminance(lux)
        /// </summary>
        public float lux { get; set; }
        /// <summary>
        /// pressure(pa)
        /// </summary>
        public float pressure { get; set; }
        /// <summary>
        /// posture (formats:"x: 1; y: - 29; z: - 2903")
        /// </summary>
        public string fPosture { get; set; }
        /// <summary>
        /// Spare battery (formats:"55,3.88,0")
        /// </summary>
        public string backBattery { get; set; }
        /// <summary>
        /// Data type 0: real-time; 1: Supplementary report; 2: Alarm
        /// </summary>
        public int reportType { get; set; }
        /// <summary>
        /// Network type 0: unknown 1:1G 2:2G 3:3G 4:4: G 5:5G Others are n
ot displayed
```

```
        /// </summary>
        public int networkType { get; set; }
    }
```

2.Alarm data

```
        [HttpPost]
        public MBackResult ReceiveAlarmData([FromBody]TerminalAlarm parm)
        {
            Log.Instance.Warn("Alarm data：" + JsonConvert.SerializeObject(
parm));
            MBackResult result = new MBackResult();
            result.Result = 200;
            return result;
        }
```

TerminalAlarm:

```
/// <summary>
    /// Alarm data
    /// </summary>
    public class TerminalAlarm
    {
        /// <summary>
        /// Equipment No
        /// </summary>
        public string assetId { get; set; }
        /// <summary>
        /// Alarm type (refer to the following description of alarm type)
        /// </summary>
        public int alarmType { get; set; }
        /// <summary>
        /// Time (UTC time)
        /// </summary>
        public string dateTime { get; set; }
        /// <summary>
        /// longitude(WGS-84)
        /// </summary>
        public double longitude { get; set; } = 0.0;
        /// <summary>
        /// latitude(WGS-84)
        /// </summary>
        public double latitude { get; set; } = 0.0;
        /// <summary>
```

```
        /// speed(km/h)
        /// </summary>
        public int speed { get; set; }
        /// <summary>
        /// mileage(km)
        /// </summary>
        public long mileage { get; set; }
        /// <summary>
        /// Cell code data is displayed in MNC, MCC, LAC and CID formats
        /// </summary>
        public string cells { get; set; }
        /// <summary>
        /// alarm describe
        /// </summary>
        public string describe { get; set; }
        /// <summary>
        /// Serial number of files
        /// </summary>
        public string fileIndex{ get; set; }
    }
```

3.Event data

```
        [HttpPost]
        public MBackResult ReceiveEventData([FromBody]TerminalEvent parm)
        {
            Log.Instance.Warn("Event data：" + JsonConvert.SerializeObject(
parm));
            MBackResult result = new MBackResult();
            result.Result = 200;
            return result;
        }
```

TerminalEvent:

```
    /// <summary>
    /// Event data
    /// </summary>
    public class TerminalEvent
    {
        /// <summary>
        /// Equipment No
        /// </summary>
        public string assetId { get; set; }
```

```
        /// <summary>
        /// Event type (1:Unlock;2:Lock;3:Sealing of exit area;;4:Unblock t
he entrance area;5:Open box(door);6:Close box(door))
        /// </summary>
        public int eventType { get; set; }
        /// <summary>
        /// Unlocking type, only for lock on/off events (1: swipe card to u
nlock 2: remote unlock 3: Bluetooth unlock 4: cut rope to unlock 5: SMS unl
ock 6: regional trigger)
        /// </summary>
        public int unLockType { get; set; }
        /// <summary>
        /// Time (UTC time)
        /// </summary>
        public string dateTime { get; set; }
        /// <summary>
        /// longitude(WGS-84)
        /// </summary>
        public double longitude { get; set; } = 0.0;
        /// <summary>
        /// latitude(WGS-84)
        /// </summary>
        public double latitude { get; set; } = 0.0;
        /// <summary>
        /// speed(km/h)
        /// </summary>
        public int speed { get; set; }
        /// <summary>
        /// mileage(km)
        /// </summary>
        public long mileage { get; set; }
        /// <summary>
        /// Cell code data is displayed in MNC, MCC, LAC and CID formats
        /// </summary>
        public string cells { get; set; }
        /// <summary>
        /// Swipe card No,only for lock products
        /// </summary>
        public string card { get; set; }
        /// <summary>
        /// unlock password,only for lock products
        /// </summary>
        public string password { get; set; }
    }
```

4.Command response data

```
        [HttpPost]
        public MBackResult ReceiveInsData([FromBody]TerminalCommand parm)
        {
            Log.Instance.Warn("Command response data：" + JsonConvert.Seria
lizeObject(parm));
            MBackResult result = new MBackResult();
            result.Result = 200;
            return result;
        }
```

TerminalCommand:

```
    /// <summary>
    /// Command response data
    /// </summary>
    public class TerminalCommand
    {
        /// <summary>
        /// Equipment No
        /// </summary>
        public string assetId { get; set; }
        /// <summary>
        /// Command type
        /// </summary>
        public string commandType { get; set; }
        /// <summary>
        /// Equipment response content
        /// </summary>
        public string content { get; set; }
        /// <summary>
        /// Time (UTC time)
        /// </summary>
        public string dateTime { get; set; }
    }
```

5、Slave Data

```
        [HttpPost]
        public MBackResult ReceiveInsData([FromBody]SlaveMachineLocation pa
rm)
        {
```

```
            Log.Instance.Warn("slave data：" + JsonConvert.SerializeObject(
parm));
            MBackResult result = new MBackResult();
            result.Result = 200;
            return result;
        }
```

SlaveMachineLocation

```
    /// <summary>
    /// slave data
    /// </summary>
    public class SlaveMachineLocation
    {
        /// <summary>
        /// Equipment No
        /// </summary>
        public string assetId { get; set; }
        /// <summary>
        /// longitude(WGS-84)
        /// </summary>
        public double longitude { get; set; } = 0.0;
        /// <summary>
        /// latitude(WGS-84)
        /// </summary>
        public double latitude { get; set; } = 0.0;
        /// <summary>
        /// speed(km/h)
        /// </summary>
        public int speed{ get; set; }
        /// <summary>
        /// direction(0~360)
        /// </summary>
        public int direction { get; set; } = 0;
        /// <summary>
        /// mileage(km)
        /// </summary>
        public long mileage { get; set; } = 0L;
        /// <summary>
        /// Positioning time ( UTC )
        /// </summary>
        public string gpsTime { get; set; }
        /// <summary>
        /// Receiving time ( UTC )
        /// </summary>
```

```
        public string recvTime { get; set; }
        /// <summary>
        /// Slave Positioning time ( UTC )
        /// </summary>
        public string subGpsTime { get; set; }
        /// <summary>
        /// Slave Power
        /// </summary>
        public int battery { get; set; }
        /// <summary>
        /// Slave Voltage
        /// </summary>
        public string voltage { get; set; }
        /// <summary>
        /// SLAVE Equipment No
        /// </summary>
        public string subAssetID { get; set; }
        /// <summary>
        /// SLAVE Type  1-JT126 , 4-JT709 , 5-JT801 , 6-JT802
        /// </summary>
        public int sensorType { get; set; }
        /// <summary>
        /// For 802 device status data (Json string), other types of device
 data can be ignored
        /// </summary>
        public string statusJson { get; set; }
        /// <summary>
        /// Slave lock status  0-Off 1-On
        /// </summary>
        public int locStatus { get; set; }
        /// <summary>
        /// Slave Rope locking status (1: pulling out 0: inserting - 1: non
e)
        /// </summary>
        public int locRope { get; set; }
        /// <summary>
        ///RSSI
        /// </summary>
        public int rssi { get; set; }
        /// <summary>
        /// Slave Temperature - 1000 means none
        /// </summary>
        public double temperature { get; set; } = -1000.0;
        /// <summary>
        /// Slave Humidity 0 means none
        /// </summary>
```

```
        public int humidity { get; set; }
        /// <summary>
        /// Slave Event -1：none 0:Lock event 1:BLE unlock event 2:Open bac
k cover waring 3:Lora unlock event 4:Lock rope cut warning 5:Key wake up ev
ent 6:Heartbeat packet 7：Charging report event 8/20:Pull out the lock rope
 event 9：RFID unlock event  10：Illegal RFID unlock warning  14：Sub lock s
ignal lost warning 15：Valve closed event 16：Valve open event 17：Low batte
ry warning 18：Anti-disassembly 19：Disassembly of electronic compartment 2
1：Lock rope insertion 22：Bluetooth connection 23：Open emergency storage w
arning 24：Close emergency storage warning 25：Valve abnormally open warnin
g 26：Turn off the knob event 27：Turn on the knob event 28:Unlocking Error
 29:Abnormal Motor  30:NFC trigger
        /// </summary>
        public int eventType { get; set; } = -1;
        /// <summary>
        /// Slave locking times
        /// </summary>
        public int locTimes { get; set; }
        /// <summary>
        /// Slave Positioning timestamp
        /// </summary>
        public long subGpsTimestamp { get; set; }
    }
```

Java

Example Description

```
Some field data may be meaningless due to different equipment types. If it is not available, it can be ignored
```

1.Positioning data

```
    @PostMapping("/receiveLocationData")
    public MBackResult receiveLocationData(@RequestHeader(value = "headerKey", required = true) String headerKey, @RequestBody TerminalLocation param) {
        log.info("ReceiveLocationData:" + JSON.toJSON(param));
        MBackResult result = new MBackResult();
        result.setResult(200);
        return result;
    }
```

TerminalLocation:

```
@Data
public class TerminalLocation {
    /**
     * Equipment No
     */
    public String assetId;
    /**
     * longitude(WGS-84)
     */
    public Double longitude = 0.0d;
    /**
     * latitude(WGS-84)
     */
    public Double latitude = 0.0d;
    /**
     * speed(km/h)
     */
    public Integer speed;
    /**
     * direction(0~360)
```

```java
     */
    public Integer direction = 0;
    /**
     * mileage(km)
     */
    public Long mileage = 0L;
    /**
     * Positioning time ( UTC )
     */
    public String gpsTime;
    /**
     * Receiving time ( UTC )
     */
    public String recvTime;
    /**
     * Positioning type (0: no positioning; 1: GPS positioning; 2: base sta
tion positioning;3:Cells positioning)
     */
    public Integer locType = 0;
    /**
     * GPRS signal
     */
    public Integer cellSignal = 0;
    /**
     * Satellite signal
     */
    public Integer gnssSignal = 0;
    /**
     * Cell code data, MNC, MCC, LAC, CID
     */
    public String cells;
    /**
     * Power
     */
    public Integer battery;
    /**
     * Voltage(V)
     */
    public String voltage;
    /**
     * Lock/vehicle status JSON
     */
    public String statusJson;
    /**
     * Extension information JSON
     */
```

```
    public String expandInfo;
}
```

Lock/vehicle status JSON

```
@Data
public class StatusJson {
     /**
      *  Lock status (0: Off 1: On)
      */
    private Integer lockStatus;
    /**
      * Rope locking status (1: pulling out 0: inserting - 1: none)
      */
    private Integer lockRope;
    /**
      * ACC status (1: On 0: Off 1: None)
      */
    private Integer acc;
    /**
      * State of oil cut-off electric switch (1: On 0: Off 1: None)
      */
    private Integer fuelCut;
    /**
      * Door opening/closing status (1: open 0: close - 1: none)
      */
    private Integer door;
    /**
      * Engine status (1: On 0: Off 1: None)
      */
    private Integer engine;
}
}
```

JSON description of extension information

```
@Data
public class ExpendInfo {
     /**
      * Temperature - 1000 means none
      */
    private String temperature;
    /**
      * Humidity 0 means none
      */
```

```
    private String humidity;
    /**
     * Spare battery (formats:"55,3.88,0")
     */
    private String backBattery;
    /**
     * Oil level value - 1 means none ("- 1, - 1, - 1")
     */
    private String fuels;
    /**
     * acceleration(formats:"x:1;y:-29;z:-2903")
     */
    private String acceleration;
    /**
     * Illuminance(lux)
     */
    private String lux;
    /**
     * pressure(pa)
     */
    private String pressure;
    /**
     * posture(formats:"x:1;y:-29;z:-2903")
     */
    private String posture;
    /**
     * Network type 0: unknown 1:1G 2:2G 3:3G 4:4: G 5:5G Others are not di
splayed
     */
    private String networkType;

    /**
     * Data type 0: real-time; 1: Supplementary report; 2: Alarm
     */
    private String reportType;
}
```

2.Alarm data

```
    @PostMapping("/receiveAlarmData")
    public MBackResult receiveAlarmData(@RequestHeader(value = "headerKey",
 required = true) String headerKey, @RequestBody TerminalAlarm param) {
        log.info("receiveAlarmData:" + JSON.toJSON(param));
        MBackResult result = new MBackResult();
        result.setResult(200);
```

```
            return result;
        }
```

TerminalAlarm：

```
@Data
public class TerminalAlarm {
    /**
     * Equipment No
     */
    public String assetId;
    /**
     * Alarm type (refer to the following description of alarm type)
     */
    public int alarmType;
    /**
     *  Time (UTC time)
     */
    public String dateTime;
    /**
     * longitude(WGS-84)
     */
    public Double longitude = 0.0d;
    /**
     * latitude(WGS-84)
     */
    public Double latitude = 0.0d;
    /**
     * speed(km/h)
     */
    public Integer speed;
    /**
     * mileage(km)
     */
    public Long mileage;
    /**
     * Cell code data is displayed in MNC, MCC, LAC and CID formats
     */
    public String cells;
    /**
     * alarm describe
     */
    public String describe;
    /**
     * Serial number of files
```

```
     */
    public String fileIndex;
}
```

3.Event data

```
    @PostMapping("/receiveEventData")
    public MBackResult receiveEventData(@RequestHeader(value = "headerKey",
 required = true) String headerKey, @RequestBody TerminalEvent param) {
        log.info("receiveEventData:" + JSON.toJSON(param));
        MBackResult result = new MBackResult();
        result.setResult(200);
        return result;
    }
```

TerminalEvent：

```
@Data
public class TerminalEvent {
    /**
     * Event data
     */
    public String assetId;
    /**
     * Event type (1:Unlock;2:Lock;3:Sealing of exit area;;4:Unblock the en
trance area;5:Open box(door);6:Close box(door))
     */
    public Integer eventType;
    /**
     * Unlocking type, only for lock on/off events (1: swipe card to unlock
 2: remote unlock 3: Bluetooth unlock 4: cut rope to unlock 5: SMS unlock 6
: regional trigger)
     */
    public Integer unLockType;
    /**
     * Time (UTC time)
     */
    public String dateTime;
    /**
     * longitude(WGS-84)
     */
    public Double longitude = 0.0d;
    /**
     * latitude(WGS-84)
```

```
     */
    public Double latitude = 0.0d;
    /**
     * speed(km/h)
     */
    public Integer speed;
    /**
     * mileage(km)
     */
    public Long mileage;
    /**
     * Cell code data is displayed in MNC, MCC, LAC and CID formats
     */
    public String cells;
    /**
     * Swipe card No,only for lock products
     */
    public String card;
    /**
     * unlock password,only for lock products
     */
    public String password;
}
```

4.Command response data

```
    @PostMapping("/receiveInsData")
    public MBackResult receiveInsData(@RequestHeader(value = "headerKey", r
equired = true) String headerKey, @RequestBody TerminalCommand param) {
        log.info("receiveInsData:" + JSON.toJSON(param));
        MBackResult result = new MBackResult();
        result.setResult(200);
        return result;
    }
```

TerminalCommand：

```
@Data
public class TerminalCommand {
    /**
     * Equipment No
     */
    public String assetId;
    /**
```

```
        * Command type
        */
       public String commandType;
       /**
        * Equipment response content
        */
       public String content;
       /**
        * Time (UTC time)
        */
       public String dateTime;
   }
```

5、Slave Data

```
    @PostMapping("/receiveLocationData")
    public MBackResult receiveLocationData(@RequestHeader(value = "headerK
ey", required = true) String headerKey, @RequestBody TerminalLocation param
) {
        log.info("ReceiveLocationData:" + JSON.toJSON(param));
        MBackResult result = new MBackResult();
        result.setResult(200);
        return result;
    }
```

SlaveMachineLocation

```
    /// <summary>
    /// slave data
    /// </summary>
    public class SlaveMachineLocation
    {
        /// <summary>
        /// Equipment No
        /// </summary>
        public String assetId { get; set; }
        /// <summary>
        /// longitude(WGS-84)
        /// </summary>
        public Double longitude { get; set; } = 0.0;
        /// <summary>
        /// latitude(WGS-84)
        /// </summary>
        public Double latitude { get; set; } = 0.0;
```

```
        /// <summary>
        /// speed(km/h)
        /// </summary>
        public Integer speed{ get; set; }
        /// <summary>
        /// direction(0~360)
        /// </summary>
        public Integer direction { get; set; } = 0;
        /// <summary>
        /// mileage(km)
        /// </summary>
        public Long mileage { get; set; } = 0L;
        /// <summary>
        /// Positioning time(UTC)
        /// </summary>
        public String gpsTime { get; set; }
        /// <summary>
        /// Receiving time(UTC)
        /// </summary>
        public String recvTime { get; set; }
        /// <summary>
        /// Slave Positioning time(UTC)
        /// </summary>
        public String subGpsTime { get; set; }
        /// <summary>
        /// Slave Power
        /// </summary>
        public Integer battery { get; set; }
        /// <summary>
        /// Slave Voltage
        /// </summary>
        public String voltage { get; set; }
        /// <summary>
        /// SLAVE Equipment No
        /// </summary>
        public string subAssetID { get; set; }
        /// <summary>
        /// SLAVE Type  1-JT126 , 4-JT709 , 5-JT801 , 6-JT802
        /// </summary>
        public Integer sensorType { get; set; }
        /// <summary>
        /// For 802 device status data (Json string), other types of device
  data can be ignored
        /// </summary>
        public string statusJson { get; set; }
        /// <summary>
```

```
        /// Slave lock status  0-Off 1-On
        /// </summary>
        public Integer locStatus { get; set; }
        /// <summary>
        /// Slave Rope locking status (1: pulling out 0: inserting - 1: non
e)
        /// </summary>
        public Integer locRope { get; set; }
        /// <summary>
        ///RSSI
        /// </summary>
        public Integer rssi { get; set; }
        /// <summary>
        /// Slave Temperature - 1000 means none
        /// </summary>
        public Double temperature { get; set; } = -1000.0;
        /// <summary>
        /// Slave Humidity 0 means none
        /// </summary>
        public Integer humidity { get; set; }
        /// <summary>
        /// Slave Event -1：none 0:Lock event 1:BLE unlock event 2:Open bac
k cover waring 3:Lora unlock event 4:Lock rope cut warning 5:Key wake up ev
ent 6:Heartbeat packet 7：Charging report event 8/20:Pull out the lock rope
 event 9：RFID unlock event  10：Illegal RFID unlock warning  14：Sub lock s
ignal lost warning 15：Valve closed event 16：Valve open event 17：Low batte
ry warning 18：Anti-disassembly 19：Disassembly of electronic compartment 2
1：Lock rope insertion 22：Bluetooth connection 23：Open emergency storage w
arning 24：Close emergency storage warning 25：Valve abnormally open warnin
g 26：Turn off the knob event 27：Turn on the knob event 28:Unlocking Error
 29:Abnormal Motor  30:NFC trigger
        /// </summary>
        public Integer eventType { get; set; } = -1;
        /// <summary>
        /// Slave unlock times
        /// </summary>
        public Integer locTimes { get; set; }
        /// <summary>
        /// Slave Positioning timestamp
        /// </summary>
        public Long subGpsTimestamp { get; set; }
    }
```

Webhook Server-Python Code

Python code example:

```
# pip install Flask
"""# webhook URL
  # 120.10.11.11 public IP address
  http://120.10.11.11:9999/lock/lockevent
  http://120.10.11.11:9999/lock/Positiondata
  http://120.10.11.11:9999/lock/alarm
  http://120.10.11.11:9999/lock/cmdresponse
"""
from flask import Flask, request

app = Flask(__name__)

@app.route('/lock/lockevent', methods=['POST'])
def lock_lockevent():
    if request.method == 'POST':
        print("Data received from Webhook is: ", request.json)
        return "Webhook received! lockevent"

@app.route('/lock/Positiondata', methods=['POST'])
def lock_Positiondata():
    if request.method == 'POST':
        print("Data received from Webhook is: ", request.json)
        return "Webhook received! Positiondata"

@app.route('/lock/alarm', methods=['POST'])
def lock_alarm():
    if request.method == 'POST':
        print("Data received from Webhook is: ", request.json)
        return "Webhook received! alarm"

@app.route('/lock/cmdresponse', methods=['POST'])
def lock_cmdresponse():
    if request.method == 'POST':
        print("Data received from Webhook is: ", request.json)
        return "Webhook received! cmdresponse"


app.run(host='0.0.0.0', port=9999)
```

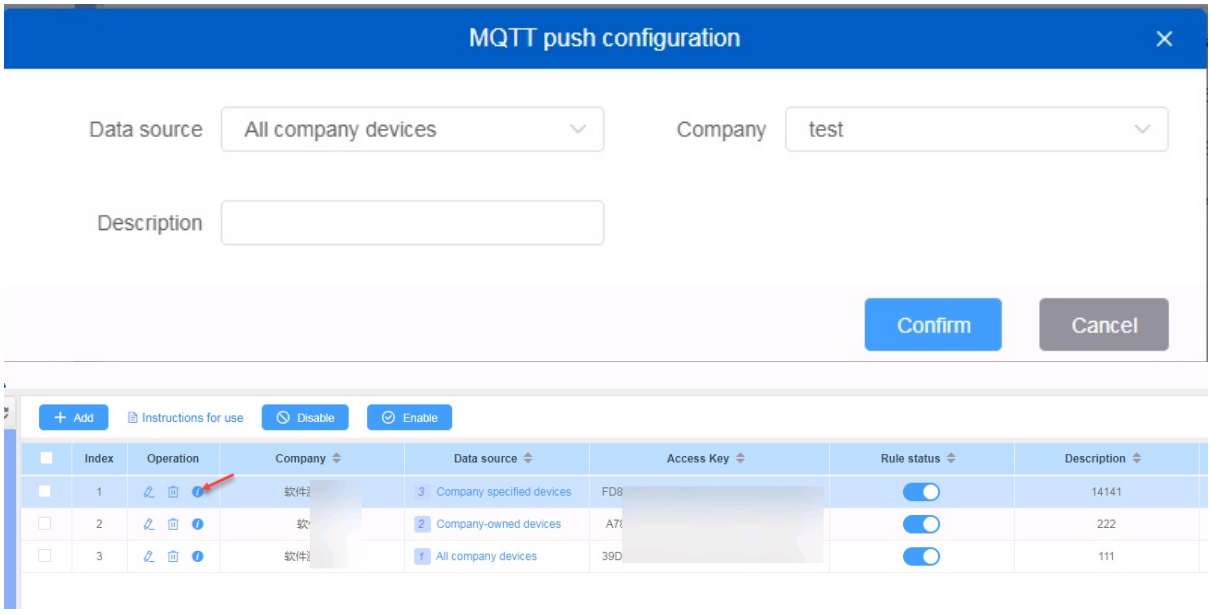
MQTT Data Push Service

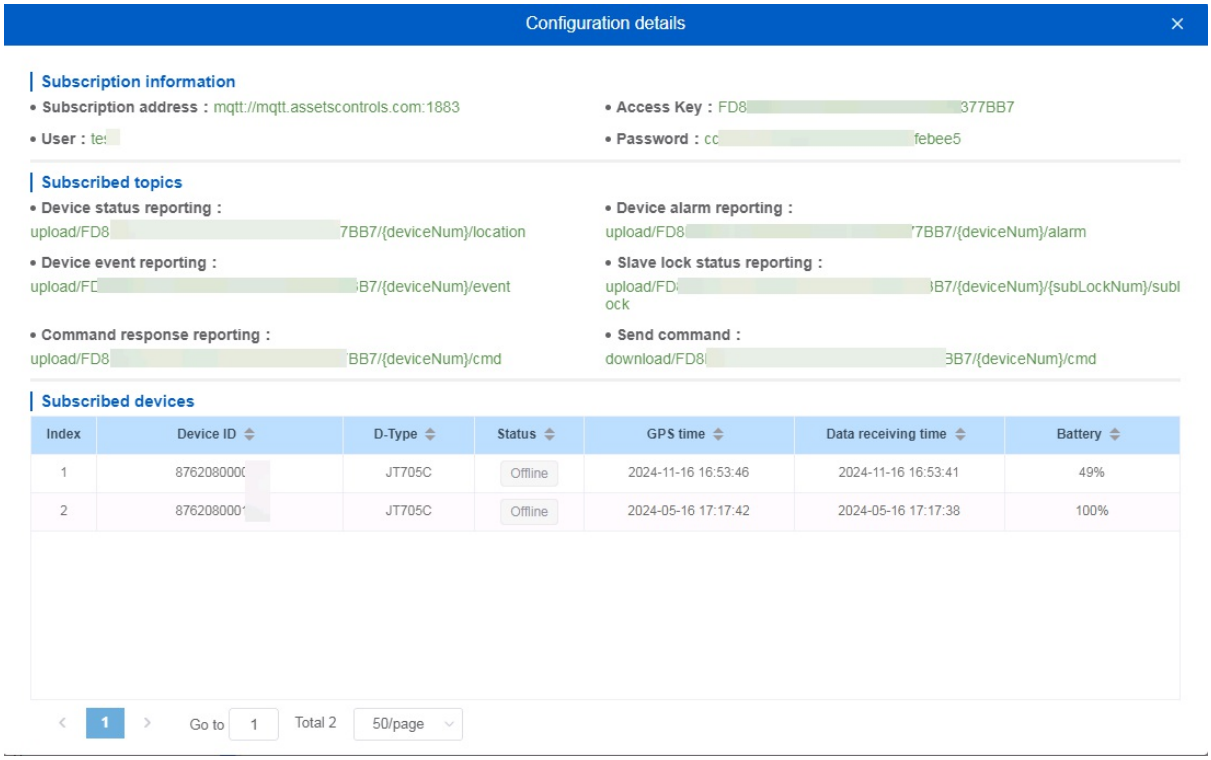
Describe

1.The MQTT push service provided by Jointech is a service that sends device data to data users in real time. When the device uploads data, the push service will sense it immediately, organize and package the data, and send it to the MQTT server. The customer data receiving service obtains data from the MQTT server, and then the customer processes the data according to his own business.
2.The MQTT rule configuration interface is mainly used to facilitate users to configure which devices to use for MQTT push services
3.Subscription topic description

MQTT Rule Configuration

In this interface, users can configure the devices that need to be pushed by MQTT. By selecting the data source, users can select the devices that need to be pushed in multiple ways.





Data source description

1.All devices of the company

```
Indicates all devices under the current company and its subsidiaries
```

2.Company-owned devices

```
Indicates the current company and all the devices under it
```

3.Company specified devices

```
Indicates the designated devices under the current company and its subsidia
ries
```

Subscription topic and payload description

Replace {accessKey} with the actual value assigned by the system and {deviceNum} with the ID of the device

Subscription topic	Description
	Subscribe to all types of data

upload/{accessKey}/#	on all devices
upload/{accessKey}/{deviceNum}/#	Subscribe to all types of data on a device
upload/{accessKey}/+/location	Subscribe to all device location data
upload/{accessKey}/{deviceNum}/location	Subscribe to a device location data
upload/{accessKey}/+/alarm	Subscribe to all device alarm data
upload/{accessKey}/{deviceNum}/alarm	Subscribe to a device alarm data
upload/{accessKey}/+/event	Subscribe to all device event data
upload/{accessKey}/{deviceNum}/event	Subscribe to a device event data
upload/{accessKey}/+/cmd	Subscribe to all device command response data
upload/{accessKey}/{deviceNum}/cmd	Subscribe to a device command response data
upload/{accessKey}/{deviceNum}/{subLockNum}/sublock	Subscribe to a slave lock(sensor) data on a device
upload/{accessKey}/{deviceNum}/+/sublock	Subscribe to all slave lock(sensor) data on a device
download/{accessKey}/{deviceNum}/cmd	Send a device command data to the terminal

1.Server address and description

Parameter name	Description
Server address	mqtt://mqtt.assetscontrols.com:1883
Protocol version	MQTT 3.1.1

QoS Level	Recommended QoS0 (at most once)

2.Client account and password

```
The login account and password (after MD5 encryption) configured for the cu
rrent MQTT rule can also be viewed directly in the details interface of the
 rule.
```

3.Subscription topic Description

3.1 Device status reporting

Topic：upload/{accessKey}/{deviceNum}/location
Description：devices positioning status data
e.g.:


```
{"assetId":"8052400203","battery":255,"cellSignal":31,"cells":"460,0,10352,
188975300","direction":0,"expandInfo":"{\"angle\":\"null\",\"backBattery\":
\"null\",\"fuels\":\"-1,-1,-1\",\"humidity\":\"0\",\"lux\":\"null\",\"netwo
rkType\":\"0\",\"pressure\":\"null\",\"reportType\":\"null\",\"temperature\
":\"-1000.0\"}","gnssSignal":0,"gpsTime":"2025-02-07T07:26:30Z","latitude":
22.6700711039462,"locType":2,"longitude":113.922999834147,"mileage":0,"recv
Time":"2025-02-07T07:25:20.832Z","speed":0,"statusJson":"{\"lockRope\":0,\"
lockStatus\":0}","voltage":"0.0"}
```


Json Field Description

Parameter name	Type	Description
assetId	String	Device ID
longitude	Double	longitude(WGS-84)
latitude	Double	latitude(WGS-84)
speed	Integer	speed(km/h)
direction	Integer	direction(0~360)
mileage	Long	mileage(km)
gpsTime	String	Positioning time（UTC）
recvTime	String	Receiving time（UTC）

recvTime	String	Receiving time（UTC）
locType	Integer	Positioning type (0: no positioning; 1: GPS positioning; 2: base station positioning;3:Cells positioning)
cellSignal	Integer	GPRS signal
gnssSignal	Integer	Satellite signal
cells	String	Cell code data is displayed in MNC, MCC, LAC and CID formats
battery	Integer	Power, 255 indicates charging
voltage	String	Voltage(V)
statusJson	String	Lock/vehicle mounted status JSON (refer to the status JSON description below)
expandInfo	String	Extension information JSON (refer to the following extension information JSON description)

Lock Status JSON Description

Parameter name	Type	Description
lockRope	Integer	Rope locking status (1: pulling out; 0: inserting; - 1: none)
lockStatus	Integer	Lock status (0: Off; 1: On)

Vehicle status JSON description

Parameter name	Type	Description
acc	Integer	Engine switch status (1: On; 0: Off; 1: None)
fuelCut	Integer	State of oil cut-off electric switch (1: On; 0: Off; 1: None)
door	Integer	Door opening/closing status (1: open; 0: close; - 1: none)
engine	Integer	Engine status (1: On; 0: Off; 1: None)

JSON description of extension information

Parameter name	Type	Description

humidity	String	Humidity 0 means none
fuels	String	Oil level value - 1 means none (“- 1, - 1, - 1”)
fAcceleration	String	acceleration(formats:“x: 1; y: - 29; z: - 2903”)
lux	float	Illuminance(lux)
pressure	float	pressure(pa)
posture	String	Posture (formats:“x: 1; y: - 29; z: - 2903”)
fVoltage	Double	Voltage value(V)
backBattery	String	Backup battery (“55,3.88,0”)
fReportType	Integer	Data type (0: real-time; 1: supplementary report; 2: alarm)
fNetworkType	Integer	Network type (0: unknown 1:1G 2:2G 3:3G 4:4G 5:5G)

3.2 Device alarm reporting

Topic：upload/{accessKey}/{deviceNum}/alarm
Description：device alarm data, such as low battery, lock rope cut alarm
e.g.:


```
{"alarmType":44,"assetId":"8454601010","cells":"460,0,9383,149428936","date
Time":"2020-04-01T00:00:25Z","describe":"","fileIndex":"1585728025","latitu
de":22.6766686619127,"longitude":113.928864975252,"mileage":59,"speed":0}
```


Json Field Description

Parameter name	Type	Description
assetId	String	Device ID
alarmType	Integer	Alarm type (refer to the following description of alarm type)
dateTime	String	Time (UTC time)
longitude	Double	longitude(WGS-84)
latitude	Double	latitude(WGS-84)
speed	Integer	speed(km/h)

mileage	Long	mileage(km)
cells	String	Cell code data is displayed in MNC, MCC, LAC and CID formats
describe	String	alarm describe
fileIndex	String	Serial number of files

Alarm Type Description：

Alarm Type	Alarm Name
1	Speeding Alarm
2	Fatigue driving
3	Danger warning
4	GNSS module failure
5	GNSS antenna not connected or cut off
6	GNSS antenna short circuit
7	device main power supply under voltage
8	Power failure of device main power supply
9	device LCD or display failure
10	TTS module failure
11	Camera fault
12	IC card module failure
13	Speeding warning
15	Power failure
19	Overtime parking
23	Route deviation alarm
24	Vehicle VSS failure

24	Vehicle VSS failure
26	Vehicle theft
27	Illegal vehicle ignition
28	Illegal vehicle displacement
29	Collision warning
30	Rollover warning
31	Illegal door opening alarm
32	Video signal loss alarm
33	Video signal blocking alarm
34	Storage unit failure alarm
35	Other video equipment fault alarm
36	Passenger train overload alarm
37	Abnormal driving behavior alarm
38	Video recording reaches storage threshold alarm
40	Lock rope cut
41	Vibration
42	Long-time unlocking
43	Unlock password error for 5 consecutive times
44	Swiping illegal RFID card
45	Low power
46	Back cover opened
47	Motor stuck
48	Enter fence alarm
49	Exit fence alarm

51	SOS
52	Towing alarm
54	Fuel level change alarm
55	Enter hot spot alarm
56	Exit hot spot alarm
57	Access road alarm
58	Exit road alarm
63	Temperature and humidity alarm
64	Forward collision alarm
65	Lane departure alarm
66	Too close distance alarm
67	Pedestrian collision alarm
68	Frequent lane change alarm
69	Road sign overrun alarm
70	Obstacle alarm
71	Road sign recognition events
72	Active capture event
73	Lane changing of solid line
74	Pedestrian monitoring in car aisle
92	Fatigue driving alarm
93	Call the police
94	Smoking alarm
95	Distracted driving alarm
96	Driver abnormal alarm

97	Auto capture events
98	Driver change event
99	Probe blocked
100	Overtime driving
101	Not wearing seat belt
102	Infrared blocking sunglasses failure
103	Hands off the steering wheel
104	Playing with phone
110	Tire pressure alarm
125	Rear approach alarm
126	Left rear approach alarm
127	Right rear approach alarm
131	Rapid acceleration
132	Emergency deceleration
133	Emergency turning
134	Idle alarm
135	Abnormal flameout
136	Neutral skid alarm
137	Engine overspeed
141	Overspeed
145	Excess load of vehicle
146	Excess load of road
151	Height limit alarm
160	Safety belt alarm

160	Safety belt alarm
161	Emergency brake alarm
162	Neutral skating
163	GPRS reconnection
164	Dispatching panel connection
165	Dispatch panel disconnected
166	CANbus disconnection alarm
167	CANbus fault code upload alarm
168	Limit drive alarm
180	Host dismantling
190	Damage of upper cover
191	GPS antenna interference
192	Low power sleep alarm
193	Device exception
194	Unlock password error
195	Do not perform unlocking without positioning
196	No unlocking outside the fence
197	Attitude alarm
198	Slave sensor signal lost
199	Supervision status forbids unlocking
200	Main battery replacement
201	Self-check exception
202	not enough storage space
203	Lock bar bent

205	lock bar missing
206	Forbidden to unlock in the sealed state
316	Light alarm
332	Pressure alarm
400	Idling alarm
402	Fence stop alarm
403	Out fence unlock
404	Departure delay warning
405	Departure delay alarm
406	Late warning
407	Late alarm

3.3 device event reporting

Topic：upload/{accessKey}/{deviceNum}/event
Description：device event data, such as unlocking and locked
e.g.:


```
{"assetId":"8454601010","card":"","cells":"460,0,9383,149428936","dateTime":
"2025-02-07T06:54:52Z","describe":"","eventType":1,"latitude":22.6766686619
127,"longitude":113.928864975252,"mileage":59,"password":"","speed":0,"unLo
ckType":2}
```


Json Field Description

Parameter name	Type	Description
assetId	String	Device ID
eventType	Integer	Event type (refer to the following description of event type)
unLockType	Integer	Unlocking type, only for lock on/off events (1: swipe card to unlock 2: remote unlock 3: Bluetooth unlock 4: cut rope to

		unlock 5: SMS unlock 6: regional trigger)
dateTime	String	Time (UTC time)
longitude	Double	longitude(WGS-84)
latitude	Double	latitude(WGS-84)
speed	Integer	speed(km/h)
mileage	Long	mileage(km)
cells	String	Cell code data is displayed in MNC, MCC, LAC and CID formats
card	String	Swipe card No,only for lock products
password	String	unlock password,only for lock products
describe	String	Description information. For route analysising waybill events 13-18, the description will have a Json string value, showing the waybill unique identifier (waybillGUID) and waybill number(Declaration No.)

3.4 Command response report

Topic：upload/{accessKey}/{deviceNum}/cmd
Description：The device responds to the commands sent by the platform
Command response description：Different commands have different response results. For specific commands and their descriptions, please refer to the protocol document.
e.g.:

```
 //command operation success response
{"assetId":"794308010642","commandType":"BASE1","content":"[\"794308010642\
",\"8\",\"001\",\"BASE\",\"1\",\"JT709A_20241115_HW-V1.2_RFID_7600_V2_8\",\
"0\",\"Jointech\",\"LE20B05SIM7600M21-A_CUS_JT\",\"898604B81022C1046673\",\
"868822046635957\",\"460\",\"0\",\"188975300\",\"10352\"]","dateTime":"2025
-02-07T03:52:39.964Z"}
//command operation failed response （Device not online）
{"assetId":"8294630003","commandType":"MQTT_CMD","content":"Device not onli
ne！","dateTime":"2025-01-07T09:49:43.158Z"}
//command operation failed response （Device has not been registered）
{"assetId":"8294630481","commandType":"MQTT_CMD","content":"Device has not
been registered！","dateTime":"2025-01-07T09:48:32.975Z"}
```

Json Field Description

Json Field Description

Parameter name	Type	Description
assetId	String	Device ID
commandType	String	Command type
content	String	Equipment response content
dateTime	String	Time (UTC time)

3.5 Slave lock(sensor) data reporting

Topic：upload/{accessKey}/{deviceNum}/{subLockNum}/sublock
Description：Slave Device positioning data, including Device ID, positioning time, longitude and latitude, speed, mileage and other information
e.g.:


```
{"assetId":"8052400203","battery":41,"direction":0,"eventType":3,"gpsTime":
"2025-02-07T07:16:07Z","humidity":0,"latitude":-0.0,"locRope":0,"locStatus"
:0,"locTimes":54,"longitude":-0.0,"recvTime":"2025-02-07T07:14:57.315Z","rs
si":15,"sensorType":4,"speed":0,"statusJson":"{\"lockRope\":0,\"gateway\":0
}","subAssetID":"E0171E0366","subGpsTime":"2025-02-07T07:15:14Z","subGpsTim
estamp":1738912514000,"temperature":-1000.0,"voltage":"3.75"}
```


JsonField Description

Parameter name	Type	Description
assetId	String	Device ID
longitude	Double	longitude(WGS-84)
latitude	Double	latitude(WGS-84)
speed	Integer	speed(km/h)
direction	Integer	direction(0~360)
mileage	Long	mileage(km)
gpsTime	String	Positioning time（UTC）
recvTime	String	Receiving time（UTC）

subGpsTime	String	Slave Positioning time（UTC））
subGpsTimestamp	Long	Slave Positioning timestamp
battery	Integer	Slave Power
voltage	String	Slave voltage (V)
sensorType	Integer	Slave Type 1-JT126 , 4-JT709 , 5-JT801 , 6-JT802
locStatus	Integer	Slave lock status 0-Off 1-On
locRope	Integer	Slave Rope locking status (1: pulling out 0: inserting - 1: none)
rssi	Integer	RSSI
humidity	Integer	Slave Temperature - 1000 means none
temperature	double	Slave Humidity 0 means none
locTimes	Integer	Slave unlock times
eventType	Integer	Slave Event -1：none 0:Lock event 1:BLE unlock event 2:Open back cover waring 3:Lora unlock event 4:Lock rope cut warning 5:Key wake up event 6:Heartbeat packet 7：Charging report event 8/20:Pull out the lock rope event 9：RFID unlock event 10：Illegal RFID unlock warning 14：Sub lock signal lost warning 15：Valve closed event 16：Valve open event 17：Low battery warning 18：Anti-disassembly 19：Disassembly of electronic compartment 21：Lock rope insertion 22：Bluetooth connection 23：Open emergency storage warning 24：Close emergency storage warning 25：Valve abnormally open warning 26：Turn off the knob event 27：Turn on the knob event 28:Unlocking Error 29:Abnormal Motor 30:NFC trigger
statusJson	String	For 802 device status data (Json string), other types of device data can be ignored

statusJson Field Description

“FStatusJson”:”{"bottomDisassembly":0,"emergencyKey":0,"lockMotor":0,"structuralDisassembly":0,"gateway":0,"lockValve":0}”

Parameter name	Type	Description

bottomDisassembly	Integer	bottom disassembly status,0:No disassembly，1:disassembly
emergencyKey	Integer	emergency key status 0：Seal up，1：Enable
lockMotor	lock motor status：0：Off，1：On	
structuralDisassembly	Integer	structural disassembly status,0:No disassembly，1:disassembly
lockValve	Integer	Lock Valve status,0：Off，1：On
lockKnob	Integer	Lock Knob status，0：Off，1：On
lockRope	Integer	Rope locking status 0：inserting；1：pulling out
gateway	Integer	Meaningless, can be ignored

3.6 Platform send command

Topic：download/{accessKey}/{deviceNum}/cmd
Description：Send command to the terminal data
command Description：Different device types have different command functions. For specific commands and their descriptions, please refer to the protocol documentation.
e.g.:

```
(794308010642,1,001,BASE,1)
```

Code Sample

Java 示例

Introducing paho dependency

```
<dependency>
    <groupId>org.eclipse.paho</groupId>
    <artifactId>org.eclipse.paho.client.mqttv3</artifactId>
    <version>1.2.0</version>
</dependency>
```

Establishing an MQTT connection

When establishing a connection, you need to use AccessId, secret, topic, and clientId, which are all assigned by the system. AccessId corresponds to the client's user, and secret corresponds to the client's password.

```
String user          = "u8vngl";//user
String password      = "xxxxxx";//password(MD5-32)
String topic         = "upload/{accessKey}/{deviceNum}/location";
//String topic = "upload/B176C0FF1DFF41D0126BFF7908C76E17/8041254836/locati
on"; //Subscribe to the location data of this device
//String topic = "upload/B176C0FF1DFF41D0126BFF7908C76E17/+/location"; //Su
bscribe to the location data of this configuration device
//String topic = "upload/B176C0FF1DFF41D0126BFF7908C76E17/#"; //Subscribe t
o the location, alarm, event and other data of this configuration device

int qos              = 1;//qos2 consumes more energy, please use 1 or 0
String broker        = "mqtt://mqtt.assetscontrols.com:1883";//mqtt server a
ddress
String clientId      = "u8vngl_consumer";  //User-defined client name
//When qos is 1 or 2, mqttclient uses
MemoryPersistence persistence = new MemoryPersistence();

try {
    MqttClient client = new MqttClient(broker, clientId, persistence);
    MqttConnectOptions connOpts = new MqttConnectOptions();
    connOpts.setUserName(user);
    connOpts.setPassword(password.toCharArray());
    //When cleanSession is false, the next time you log in with the same cl
ientId, you will    // be able to retrieve all stored messages
   //If true, you will retrieve the last message marked as retained
   //Set cleanSession to true during debugging and testing, and false durin
g production
    connOpts.setCleanSession(false);
    connOpts.setAutomaticReconnect(true);//Set up automatic reconnection
    client.setCallback(new MqttCallback());//Get subscription messages
    client.connect(connOpts);
    client.subscribe(topic, qos);//Subscribe the topic
} catch(MqttException me) {
    me.printStackTrace();
}
```

Get subscription messages

Paho gets messages by implementing the MqttCallback interface, which gets messages through the messageArrived method

```
@Override
public void messageArrived(String topic, MqttMessage message) throws Except
ion {
    //Please put the processing of /message into other threads. If too much
 time is spent in this method, it will affect the response of qos 1 or 2, m
aking the server think that the message is not successfully delivered.
    System.out.println("topic:" + topic + " msg:" + message);
}


/**
 * [Important Tips]
 * Processing logic after connection loss
 * 1、Reconnect
 * 2、Resubscribe to Topic Data
 * @param cause
 */
@Override
public void connectionLost(Throwable cause) {
    cause.printStackTrace();
    while (true) {
        try {
            //Reconnect
            if (!client.isConnected()) {
                client.reconnect();
            }
            //Resubscribe to Topic Data
            if (client.isConnected()) {
                client.subscribe(topic, qos);
                System.out.println("has resubscribed");
                break;
            }
            TimeUnit.SECONDS.sleep(100);
        } catch (MqttException | InterruptedException e1) {
            e1.printStackTrace();
        }
        System.out.println("client not connect");
    }
}
```

.Net 示例

```
using System;
using System.Threading;
```

```
using uPLibrary.Networking.M2Mqtt;
using uPLibrary.Networking.M2Mqtt.Messages;

class MqttClientExample
{
    private MqttClient client;
    private string broker;
    private int port;
    private string topic;
    private string clientId;
    private string username;
    private string password;
    private byte qos;
    private bool isReconnecting = false;

    // Constructor to initialize MQTT connection parameters
    public MqttClientExample(string broker, int port, string topic, string
clientId, string username, string password, byte qos)
    {
        this.broker = broker;
        this.port = port;
        this.topic = topic;
        this.clientId = clientId;
        this.username = username;
        this.password = password;
        this.qos = qos;
    }

    // Method to connect to the MQTT server
    public void Connect()
    {
        try
        {
            // Create an MQTT client instance
            client = new MqttClient(broker, port, false, null, null, MqttSs
lProtocols.None);

            // Subscribe to the connection closed event
            client.ConnectionClosed += Client_ConnectionClosed;

            // Connect to the MQTT server
            client.Connect(clientId, username, password);

            if (client.IsConnected)
            {
                Console.WriteLine("Successfully connected to the MQTT serve
```

```
r.");

                // Subscribe to the specified topic
                client.Subscribe(new string[] { topic }, new byte[] { qos }
);

                // Handle received messages
                client.MqttMsgPublishReceived += Client_MqttMsgPublishRecei
ved;

                Console.WriteLine("Subscribed to topic: " + topic);
                Console.WriteLine("Waiting for messages...");
            }
            else
            {
                Console.WriteLine("Failed to connect to the MQTT server.");
            }
        }
        catch (Exception ex)
        {
            Console.WriteLine("Error occurred while connecting: " + ex.Mess
age);
            TryReconnect();
        }
    }

    // Method to disconnect from the MQTT server
    public void Disconnect()
    {
        if (client != null && client.IsConnected)
        {
            try
            {
                // Unsubscribe from the topic
                client.Unsubscribe(new string[] { topic });

                // Disconnect from the MQTT server
                client.Disconnect();
                Console.WriteLine("Disconnected from the MQTT server.");
            }
            catch (Exception ex)
            {
                Console.WriteLine("Error occurred while disconnecting: " +
ex.Message);
            }
        }
```

```
    }

    // Method to attempt to reconnect to the MQTT server
    private void TryReconnect()
    {
        if (isReconnecting) return;
        isReconnecting = true;

        new Thread(() =>
        {
            while (!client.IsConnected)
            {
                try
                {
                    Console.WriteLine("Trying to reconnect to the MQTT serv
er...");
                    client.Connect(clientId, username, password);
                    if (client.IsConnected)
                    {
                        Console.WriteLine("Reconnected successfully.");
                        // Resubscribe to the topic
                        client.Subscribe(new string[] { topic }, new byte[]
 { qos });
                        isReconnecting = false;
                        break;
                    }
                }
                catch (Exception ex)
                {
                    Console.WriteLine("Reconnection failed: " + ex.Message)
;
                }
                Thread.Sleep(5000); // Try to reconnect every 5 seconds
            }
        }).Start();
    }

    // Event handler for the connection closed event
    private void Client_ConnectionClosed(object sender, EventArgs e)
    {
        Console.WriteLine("The connection to the MQTT server has been close
d.");
        TryReconnect();
    }

    // Event handler for received messages
```

```
    private void Client_MqttMsgPublishReceived(object sender, MqttMsgPublis
hEventArgs e)
    {
        // Process the received message
        string message = System.Text.Encoding.UTF8.GetString(e.Message);
        Console.WriteLine("Received message - Topic: " + e.Topic + ", Messa
ge content: " + message);
    }
}

class Program
{
    static void Main()
    {
        // MQTT server address
        string broker = "mqtt.assetscontrols.com";
        // MQTT server port
        int port = 1883;
        // Topic to subscribe to , {accessKey}.need to edit it.
        string topic = "upload/ {accessKey}/+/#";
        // Client ID  any Custom Name
        string clientId = "39440B3BD98045AC99EC2CF93EB15461_consumer01";
        // Username web applicaton  loginuser
        string username = "XXXXX";
        // Password  web applicaton  loginpass -MD532
        string password = "XXXXX";
        // QoS level
        byte qos = 1;

        MqttClientExample mqttClient = new MqttClientExample(broker, port,
topic, clientId, username, password, qos);
        mqttClient.Connect();

        Console.WriteLine("Press any key to disconnect...");
        Console.ReadKey();
        mqttClient.Disconnect();
    }
}
```

MQTT Client code example

.NET(C#)

Java

Python

.NET(C#)

.Net Example

```
using System;
using System.Threading;
using uPLibrary.Networking.M2Mqtt;
using uPLibrary.Networking.M2Mqtt.Messages;

class MqttClientExample
{
    private MqttClient client;
    private string broker;
    private int port;
    private string topic;
    private string clientId;
    private string username;
    private string password;
    private byte qos;
    private bool isReconnecting = false;

    // Constructor to initialize MQTT connection parameters
    public MqttClientExample(string broker, int port, string topic, string
clientId, string username, string password, byte qos)
    {
        this.broker = broker;
        this.port = port;
        this.topic = topic;
        this.clientId = clientId;
        this.username = username;
        this.password = password;
        this.qos = qos;
    }

    // Method to connect to the MQTT server
    public void Connect()
    {
        try
        {
            // Create an MQTT client instance
            client = new MqttClient(broker, port, false, null, null, MqttSs
lProtocols.None);
```

```
            // Subscribe to the connection closed event
            client.ConnectionClosed += Client_ConnectionClosed;

            // Connect to the MQTT server
            client.Connect(clientId, username, password);

            if (client.IsConnected)
            {
                Console.WriteLine("Successfully connected to the MQTT serve
r.");

                // Subscribe to the specified topic
                client.Subscribe(new string[] { topic }, new byte[] { qos }
);

                // Handle received messages
                client.MqttMsgPublishReceived += Client_MqttMsgPublishRecei
ved;

                Console.WriteLine("Subscribed to topic: " + topic);
                Console.WriteLine("Waiting for messages...");
            }
            else
            {
                Console.WriteLine("Failed to connect to the MQTT server.");
            }
        }
        catch (Exception ex)
        {
            Console.WriteLine("Error occurred while connecting: " + ex.Mess
age);
            TryReconnect();
        }
    }

    // Method to disconnect from the MQTT server
    public void Disconnect()
    {
        if (client != null && client.IsConnected)
        {
            try
            {
                // Unsubscribe from the topic
                client.Unsubscribe(new string[] { topic });

                // Disconnect from the MQTT server
```

```
                client.Disconnect();
                Console.WriteLine("Disconnected from the MQTT server.");
            }
            catch (Exception ex)
            {
                Console.WriteLine("Error occurred while disconnecting: " +
ex.Message);
            }
        }
    }

    // Method to attempt to reconnect to the MQTT server
    private void TryReconnect()
    {
        if (isReconnecting) return;
        isReconnecting = true;

        new Thread(() =>
        {
            while (!client.IsConnected)
            {
                try
                {
                    Console.WriteLine("Trying to reconnect to the MQTT serv
er...");
                    client.Connect(clientId, username, password);
                    if (client.IsConnected)
                    {
                        Console.WriteLine("Reconnected successfully.");
                        // Resubscribe to the topic
                        client.Subscribe(new string[] { topic }, new byte[]
 { qos });
                        isReconnecting = false;
                        break;
                    }
                }
                catch (Exception ex)
                {
                    Console.WriteLine("Reconnection failed: " + ex.Message)
;
                }
                Thread.Sleep(5000); // Try to reconnect every 5 seconds
            }
        }).Start();
    }
```

```
    // Event handler for the connection closed event
    private void Client_ConnectionClosed(object sender, EventArgs e)
    {
        Console.WriteLine("The connection to the MQTT server has been close
d.");
        TryReconnect();
    }

    // Event handler for received messages
    private void Client_MqttMsgPublishReceived(object sender, MqttMsgPublis
hEventArgs e)
    {
        // Process the received message
        string message = System.Text.Encoding.UTF8.GetString(e.Message);
        Console.WriteLine("Received message - Topic: " + e.Topic + ", Messa
ge content: " + message);
    }
}

class Program
{
    static void Main()
    {
        // MQTT server address
        string broker = "mqtt.assetscontrols.com";
        // MQTT server port
        int port = 1883;
        // Topic to subscribe to , {accessKey}.need to edit it.
        string topic = "upload/ {accessKey}/+/#";
        // Client ID  any Custom Name
        string clientId = "39440B3BD98045AC99EC2CF93EB15461_consumer01";
        // Username web applicaton  loginuser
        string username = "XXXXX";
        // Password  web applicaton  loginpass -MD532
        string password = "XXXXX";
        // QoS level
        byte qos = 1;

        MqttClientExample mqttClient = new MqttClientExample(broker, port,
topic, clientId, username, password, qos);
        mqttClient.Connect();

        Console.WriteLine("Press any key to disconnect...");
        Console.ReadKey();
        mqttClient.Disconnect();
    }
```

.NET(C#)

```
}
```

Java

Java Example

Introducing paho dependency

```
<dependency>
    <groupId>org.eclipse.paho</groupId>
    <artifactId>org.eclipse.paho.client.mqttv3</artifactId>
    <version>1.2.0</version>
</dependency>
```

Establishing an MQTT connection
When establishing a connection, you need to use AccessId, secret, topic, and clientId, which are all assigned by the system. AccessId corresponds to the client's user, and secret corresponds to the client's password.

```
String user          = "u8vngl";//user
String password      = "xxxxxx";//password(MD5-32)
String topic         = "upload/{accessKey}/{deviceNum}/location";
//String topic = "upload/B176C0FF1DFF41D0126BFF7908C76E17/8041254836/locati
on"; //Subscribe to the location data of this device
//String topic = "upload/B176C0FF1DFF41D0126BFF7908C76E17/+/location"; //Su
bscribe to the location data of this configuration device
//String topic = "upload/B176C0FF1DFF41D0126BFF7908C76E17/#"; //Subscribe t
o the location, alarm, event and other data of this configuration device

int qos              = 1;//qos2 consumes more energy, please use 1 or 0
String broker        = "mqtt://mqtt.assetscontrols.com:1883";//mqtt server a
ddress
String clientId      = "u8vngl_consumer";  //User-defined client name
//When qos is 1 or 2, mqttclient uses
MemoryPersistence persistence = new MemoryPersistence();

try {
    MqttClient client = new MqttClient(broker, clientId, persistence);
    MqttConnectOptions connOpts = new MqttConnectOptions();
    connOpts.setUserName(user);
    connOpts.setPassword(password.toCharArray());
    //When cleanSession is false, the next time you log in with the same cl
ientId, you will    // be able to retrieve all stored messages
   //If true, you will retrieve the last message marked as retained
   //Set cleanSession to true during debugging and testing, and false durin
```

```
g production
    connOpts.setCleanSession(false);
    connOpts.setAutomaticReconnect(true);//Set up automatic reconnection
    client.setCallback(new MqttCallback());//Get subscription messages
    client.connect(connOpts);
    client.subscribe(topic, qos);//Subscribe the topic
} catch(MqttException me) {
    me.printStackTrace();
}
```

Get subscription messages
Paho gets messages by implementing the MqttCallback interface, which gets messages through the messageArrived method

```
@Override
public void messageArrived(String topic, MqttMessage message) throws Except
ion {
    //Please put the processing of /message into other threads. If too much
 time is spent in this method, it will affect the response of qos 1 or 2, m
aking the server think that the message is not successfully delivered.
    System.out.println("topic:" + topic + " msg:" + message);
}
```

```
Java 示例
Introducing paho dependency

<dependency>
    <groupId>org.eclipse.paho</groupId>
    <artifactId>org.eclipse.paho.client.mqttv3</artifactId>
    <version>1.2.0</version>
</dependency>
Establishing an MQTT connection
When establishing a connection, you need to use AccessId, secret, topic, an
d clientId, which are all assigned by the system. AccessId corresponds to t
he client’s user, and secret corresponds to the client’s password.

String user         = "u8vngl";//user
String password     = "xxxxxx";//password(MD5-32)
String topic        = "upload/{accessKey}/{deviceNum}/location";
//String topic = "upload/B176C0FF1DFF41D0126BFF7908C76E17/8041254836/locati
on"; //Subscribe to the location data of this device
//String topic = "upload/B176C0FF1DFF41D0126BFF7908C76E17/+/location"; //Su
bscribe to the location data of this configuration device
```

```
//String topic = "upload/B176C0FF1DFF41D0126BFF7908C76E17/#"; //Subscribe t
o the location, alarm, event and other data of this configuration device

int qos             = 1;//qos2 consumes more energy, please use 1 or 0
String broker       = "mqtt://mqtt.assetscontrols.com:1883";//mqtt server a
ddress
String clientId     = "u8vngl_consumer";  //User-defined client name
//When qos is 1 or 2, mqttclient uses
MemoryPersistence persistence = new MemoryPersistence();

try {
    MqttClient client = new MqttClient(broker, clientId, persistence);
    MqttConnectOptions connOpts = new MqttConnectOptions();
    connOpts.setUserName(user);
    connOpts.setPassword(password.toCharArray());
    //When cleanSession is false, the next time you log in with the same cl
ientId, you will    // be able to retrieve all stored messages
   //If true, you will retrieve the last message marked as retained
   //Set cleanSession to true during debugging and testing, and false durin
g production
    connOpts.setCleanSession(false);
    connOpts.setAutomaticReconnect(true);//Set up automatic reconnection
    client.setCallback(new MqttCallback());//Get subscription messages
    client.connect(connOpts);
    client.subscribe(topic, qos);//Subscribe the topic
} catch(MqttException me) {
    me.printStackTrace();
}
Get subscription messages
Paho gets messages by implementing the MqttCallback interface, which gets m
essages through the messageArrived method

@Override
public void messageArrived(String topic, MqttMessage message) throws Except
ion {
    //Please put the processing of /message into other threads. If too much
 time is spent in this method, it will affect the response of qos 1 or 2, m
aking the server think that the message is not successfully delivered.
    System.out.println("topic:" + topic + " msg:" + message);
}


/**
 * [Important Tips]
 * Processing logic after connection loss
 * 1、Reconnect
```

Java

```
     * 2、Resubscribe to Topic Data
     * @param cause
     */
    @Override
    public void connectionLost(Throwable cause) {
        cause.printStackTrace();
        while (true) {
            try {
                //Reconnect
                if (!client.isConnected()) {
                    client.reconnect();
                }
                //Resubscribe to Topic Data
                if (client.isConnected()) {
                    client.subscribe(topic, qos);
                    System.out.println("has resubscribed");
                    break;
                }
                TimeUnit.SECONDS.sleep(100);
            } catch (MqttException | InterruptedException e1) {
                e1.printStackTrace();
            }
            System.out.println("client not connect");
        }
    }
```

Python

Python Example

```
# python 3.x
# pip3 install paho-mqtt
# Refer to  https://github.com/emqx/MQTT-Client-Examples/blob/master/mqtt-client-Python3/sub_tcp.py


import logging
import random
import time
from paho.mqtt import client as mqtt_client


BROKER = 'mqtt.assetscontrols.com'
PORT = 1883
TOPIC = "upload/{accessKey}/+/#"
# e.g.
#  TOPIC = "upload/B176C0FF1DFF41D488FBFF7908C76E17/+/location"
# B176C0FF1DFF41D488FBFF7908C76E17  is the {accessKey}.need to edit it.

CLIENT_ID = 'B176C0FF1DFF41D488FBFF7908C76E17_webloginuser'  #  any Custom Name
USERNAME = 'webloginuser'  # web applicaton  login-user
PASSWORD = 'cd01ab1bc1f444f6e6cb86505b00fea0'   #web applicaton  login-pass (MD532)
qos = 1
FIRST_RECONNECT_DELAY = 1
RECONNECT_RATE = 2
MAX_RECONNECT_COUNT = 12
MAX_RECONNECT_DELAY = 60

FLAG_EXIT = False


def on_connect(client, userdata, flags, rc, properties=None):
    if rc == 0 and client.is_connected():
        print("Connected to MQTT Broker!")
        client.subscribe(TOPIC,qos)
    else:
        print(f'Failed to connect, return code {rc}')
```

```python
def on_disconnect(client, userdata, rc, properties=None):
    logging.info("Disconnected with result code: %s", rc)
    reconnect_count, reconnect_delay = 0, FIRST_RECONNECT_DELAY
    while reconnect_count < MAX_RECONNECT_COUNT:
        logging.info("Reconnecting in %d seconds...", reconnect_delay)
        time.sleep(reconnect_delay)

        try:
            client.reconnect()
            logging.info("Reconnected successfully!")
            return
        except Exception as err:
            logging.error("%s. Reconnect failed. Retrying...", err)

        reconnect_delay *= RECONNECT_RATE
        reconnect_delay = min(reconnect_delay, MAX_RECONNECT_DELAY)
        reconnect_count += 1
    logging.info("Reconnect failed after %s attempts. Exiting...", reconnec
t_count)
    global FLAG_EXIT
    FLAG_EXIT = True


def on_message(client, userdata, msg):
    print(f'Received: {msg.payload.decode()}\nfrom topic: {msg.topic} \n')


def connect_mqtt():
    # client = mqtt_client.Client(CLIENT_ID)
    client = mqtt_client.Client(client_id=CLIENT_ID, callback_api_version=m
qtt_client.CallbackAPIVersion.VERSION2)
    client.username_pw_set(USERNAME, PASSWORD)
    client.on_connect = on_connect
    client.on_message = on_message
    client.connect(BROKER, PORT, keepalive=60)
    client.on_disconnect = on_disconnect
    return client

################################################################################
################################

#  Tail recursion optimization, decorators
import sys
```

```
class TailRecurseException(BaseException):
    def __init__(self, args, kwargs):
        self.args = args
        self.kwargs = kwargs

def tail_call_optimized(g):
    """
    This function decorates a function with tail call
    optimization. It does this by throwing an exception
    if it is it's own grandparent, and catching such
    exceptions to fake the tail call optimization.

    This function fails if the decorated
    function recurses in a non-tail context.
    """
    def func(*args, **kwargs):
        f = sys._getframe()
        if f.f_back and f.f_back.f_back \
            and f.f_back.f_back.f_code == f.f_code:
            raise TailRecurseException(args, kwargs)
        else:
            while 1:
                try:
                    return g(*args, **kwargs)
                except TailRecurseException as e:
                    args = e.args
                    kwargs = e.kwargs
    func.__doc__ = g.__doc__
    return func
################################################################################
###################################


# Automatic reconnection mechanism. Improve the network physical interrupti
on and the MQTT server actively disconnecting the client connection.
# Because it will not enter the on_disconnect function, the program will di
rectly exit with an error.
@tail_call_optimized  #  Tail recursion optimization, decorators
def run(try_times):
    logging.basicConfig(format='%(asctime)s - %(levelname)s: %(message)s',
                        level=logging.DEBUG)
    try:
        client = connect_mqtt()
        client.loop_forever()
    except:
        try_times = try_times + 1
```

```
            time.sleep(3)
            print("Retry_times---",try_times)
            run(try_times)
        print("Break from MQTT Broker!after trying more than {} times".format(t
ry_times))

# Initialization Attempts
#  The maximum number of attempts was 1037, and the error was "Fatal Python
 error: Cannot recover from stack overflow."
# Too many recursive function calls in Python caused stack overflow
try_times = 0



if __name__ == '__main__':
    run(try_times)
```