

目 录

数据推送服务说明

Webhook客户端代码示例

 C#(.Net)

 Java

 Webhook Server-Python

MQTT规则配置说明

MQTT客户端代码示例

 C#(.Net)

 Java

 Python

数据推送服务说明

描述

数据推送服务主要是满足客户需要设备基本数据进行二次开发的需求。该功能能够实现设备位置数据，报警数据，事件数据，离线/在线状态数据，指令下发数据的主动推送。

1. 首先需要编写一个基于POST方法的Webhook服务端接口，用来接收服务器转发过去的的数据，可以参考客户端代码示例
2. 将编写好的WEBAPI地址通过规则配置—>新建规则，填入到客户端地址里面，对应的是接口的完整URL，比如：`http://47.112.122.222:8080/revTerminalLocation`
3. 规则配置好后，预计5分钟内将会自动生效，这时候设备上报数据，系统的推送引擎就会将数据转发至配置的客户端地址。

规则配置

用户可以规则配置界面中，配置指定公司需要推送的数据，用户需要配置数据类型和客户端地址等信息

如果有消息头限制，需要加上消息头配置，没有的话，可忽略

新增规则

公司名称: TRANSPORTE SCHURTEI

* 规则名称: [输入框]

* 数据类型: 请选择

* 客户端地址: [输入框] 客户端示例

添加消息头: + 添加

描述: 请输入内容

同步到子公司: [开关]

取消 确定



数据类型说明

1. 定位数据

设备的定位数据，包括设备号，定位时间，经纬度，速度，里程等信息

JSON字符串示例

```
{
  "assetId": "8022300001",
  "longitude": 0.0,
  "latitude": 0.0,
  "speed": 0,
  "direction": 0,
  "mileage": 16,
  "gpsTime": "2022-09-01T08:59:56Z",
  "ksInfo": "",
  "recvTime": "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字段说明

参数名	类型	说明
assetId	String	设备号
longitude	Double	经度(WGS-84)
latitude	Double	纬度(WGS-84)
speed	Integer	速度(km/h)
direction	Integer	方向(0~360)
mileage	Long	里程(km)

gpsTime	String	定位时间（UTC时间）
recvTime	String	接收时间（UTC时间）
locType	Integer	定位类型（0：不定位；1：GPS定位；2：基站定位）
cellSignal	Integer	GPRS信号
gnssSignal	Integer	卫星信号
cells	String	小区码数据，以 MNC, MCC,LAC,CID的格式显示
battery	Integer	电量，255表示充电中
voltage	String	电压 (V)
statusJson	String	锁/车载状态JSON（可参考下面的状态JSON说明）
expandInfo	String	拓展信息JSON（可参考下面的扩展信息JSON说明）
ksInfo	String	KS20A设备信息JSON（可参考下面的KS20A设备信息JSON说明）， 例:,"ksInfo":{"angVelocity":21000.0,"eleStatus":0,"eleValue":3.48,"hur {"112408000035":{"dateTime":"2025-05- 14T09:39:03Z","humidity":60.0,"lux":0.0,"subId":"112408000035","ter

锁状态JSON说明

参数名	类型	说明
lockRope	Integer	锁绳状态(1: 拔出 0:插入 -1:无)
lockStatus	Integer	锁状态（0：关 1：开）

车载状态JSON说明

参数名	类型	说明
acc	Integer	引擎开关状态（1：开 0：关 -1：无）
fuelCut	Integer	断油电开关状态（1：开 0：关 -1：无）
door	Integer	开关门状态（1：开 0：关 -1：无）
engine	Integer	发动机状态（1：开 0：关 -1：无）

拓展信息JSON说明

参数名	类型	说明
temperature	String	温度 -1000表示无,单位°C
humidity	String	湿度,单位:%
fuels	String	油位值 -1表示无 (“-1, -1, -1”)
fAcceleration	String	加速度(“x:1;y:-29;z:-2903”)
lux	float	光照度,单位:lux
pressure	float	气压(pa)
posture	String	姿态 (“x:1;y:-29;z:-2903”)
fVoltage	Double	电压值,单位:V
backBattery	String	备用电池 (“55,3.88,0”)
fReportType	Integer	数据类型 (0: 实时; 1:补报; 2: 报警)
fNetworkType	Integer	网络类型 (0: 未知 1: 1G 2: 2G 3:3G 4:4G 5:5G)

KS20A设备信息JSON说明

参数名	类型	说明
temperature	double	板载温度 -1000表示无效,单位°C
humidity	Integer	湿度,单位:%
volStatus	Integer	电压状态 (0: LED电压正常 1: LED电压过压 2:电压
volValue	double	电压值,单位:V
eleStatus	Integer	电流状态 0: LED电流正常 1: LED电流过流 2:电流
eleValue	double	电流值,单位:A
pressure	float	气压(pa)
shakeStatus	Integer	0:静止 1: 摇晃
angVelocity	Long	角速度,单位:mdps

posture	String	姿态x,y,z轴值
vibrate	Integer	震幅,单位:mg
subLockId	String	锁控制器ID
subLargeCharId	String	大字控制器ID
luminance1	Integer	白灯亮度,单位:%
luminance2	Integer	红灯亮度,单位:%
luminance3	Integer	黄灯亮度,单位:%
luminance4	Integer	蓝灯亮度,单位:%
subInfo	Map<String,Object>	外设信息,可参数值:{"112408000042":{"dateTime":"13T09:55:25Z","humidity":54.0,"lux":0.0,"subId":"11

subInfo中Object说明

参数名	类型	说明
subId	String	外设ID
dateTime	String	外设上线时间,UTC时间
temperature	double	温度,单位:°C
humidity	Integer	湿度,单位:%
lux	double	光照度,单位:lux

2.报警数据

设备产生的报警数据，包括设备号，报警类型，报警时间，经纬度等信息

推送JSON示例

```
{ "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", "describe": "", "fileIndex": "1585699212" }
```

Json字段说明

参数名	类型	说明
assetId	String	设备号
alarmType	Integer	报警类型（可参考下面的报警类型说明）
dateTime	String	时间（UTC时间）
longitude	Double	经度(WGS-84)
latitude	Double	纬度(WGS-84)
speed	Integer	速度(km/h)
mileage	Long	里程(km)
cells	String	小区码数据，以 MNC, MCC,LAC,CID的格式显示
describe	String	描述
fileIndex	String	附件流水号

报警类型说明：

报警类型	报警名称
1	超速报警
2	疲劳驾驶
3	危险预警
4	GNSS模块发生故障
5	GNSS天线未接或被剪断
6	GNSS天线短路
7	终端主电源欠压
8	终端主电源掉电
9	终端LCD或显示器故障
10	TTS模块故障
11	摄像头故障

12	道路运输证IC卡模块故障
13	超速预警
15	断电报警
19	超时停车
23	路线偏离报警
24	车辆VSS故障
26	车辆被盗
27	车辆非法点火
28	车辆非法位移
29	碰撞预警
30	侧翻预警
31	非法开门报警
32	视频信号丢失报警
33	视频信号遮挡报警
34	存储单元故障报警
35	其他视频设备故障报警
36	客车超员报警
37	异常驾驶行为报警
38	特殊报警录像达到存储阈值报警
40	锁绳剪断
41	震动
42	长时间开锁
43	开锁密码连续5次错误
44	刷非法卡

45	低电量
46	开后盖
47	卡锁
48	进区域报警
49	出区域报警
50	启用后备电池
51	SOS
52	拖吊报警
54	油位报警
55	进热点报警
56	出热点报警
57	进道路报警
58	出道路报警
63	温湿度报警
64	前向碰撞报警
65	车道偏离报警
66	车距过近报警
67	行人碰撞报警
68	频繁变道报警
69	道路标识超限报警
70	障碍物报警
71	道路标志识别事件
72	主动抓拍事件
73	实线变道报警

74	车厢过道行人监测报警
92	疲劳驾驶报警
93	接打电话报警
94	抽烟报警
95	分神驾驶报警
96	驾驶员异常报警
97	自动抓拍事件
98	驾驶员变更事件
99	探头遮挡报警
100	超时驾驶报警
101	未系安全带报警
102	红外阻断型墨镜失效报警
103	双脱把报警
104	玩手机报警
110	胎压报警
125	后方接近报警
126	左侧后方接近报警
127	右侧后方接近报警
131	急加速报警
132	急减速报警
133	急转弯报警
134	怠速报警
135	异常熄火报警
136	空挡滑行报警

137	发动机超转报警
141	超速报警
145	超过车辆额定载重报警
146	超过道路承重报警
151	限高报警
160	安全带报警
161	紧急刹车报警
162	空挡滑行
163	GPRS重连
164	调度屏连接
165	调度屏断开
166	CANBUS断开连接报警
167	CANBUS故障码上传报警
168	限制开车报警
180	主机拆卸
190	上盖破坏
191	Gps天线干扰
192	低电量休眠报警
193	锁异常
194	开锁密码错误
195	没有定位不执行开锁
196	围栏外禁止开锁
197	姿态报警
198	从机信号丢失

199	监管状态禁止开锁
200	主电池更换
201	自检异常
202	存储空间不足
203	锁条弯曲
204	锁条锯断
205	锁条丢失
206	施封状态禁止开锁
216	摇晃
316	光感报警
332	气压报警
400	超时怠速报警
402	围栏超时停留报警
403	出围栏未上锁报警
404	晚发预警
405	晚发报警
406	晚到预警
407	晚到报警
1000	自定义报警

3.事件数据

设备产生的事件数据，包括设备号，事件类型，时间，经纬度等信息

推送JSON示例

```
{"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": "", "fileIndex": "1767002445", "describe": "{ \"waybillGUID\": \"190db1a2-3df1-4857-b94e-ee5937334409\", \"waybillNo\": \"45435345\" } }
```

Json字段说明

参数名	类型	说明
assetId	String	设备号
eventType	Integer	事件类型（可参考下面的事件类型说明）
unLockType	Integer	开锁类型，只针对开关锁事件(-3:应急仓关 -2:拍照上锁 -1:关锁 1: 刷卡开锁 2: 远程开锁 3: 蓝牙开锁 4: 剪绳开锁 5: 短信开锁 6: 区域触发)
dateTime	String	时间（UTC时间）
longitude	Double	经度(WGS-84)
latitude	Double	纬度(WGS-84)
speed	Integer	速度(km/h)
mileage	Long	里程(km)
cells	String	小区码数据，以 MNC, MCC,LAC,CID的格式显示
card	String	刷卡开锁卡号,仅对锁类产品有效
password	String	开锁密码,仅对锁类产品有效
describe	String	描述信息，针对运单事件13-18事件，该描述会有Json字符串的值，展示运单唯一标识（waybillGUID）和运单号（waybillNo）信息
fileIndex	String	附件流水号

事件类型说明：

事件类型	事件名称
1	开锁
2	关锁

3	出区域施封
4	进区域解封
5	开箱/门
6	关箱/门
7	蓝牙施封
8	蓝牙解封
9	远程施封
10	远程解封
11	抓拍
12	定时拍
13	离开起点
14	到达目的地
15	离开目的地
16	完成运单
17	进入途经点
18	离开途经点
19	拔出锁绳/按下开锁按键(JT705A/JT705C)
20	进入起点
10000	自定义事件

4.指令应答数据

设备产生的指令应答数据，包括设备号，指令类型，时间，指令返回内容信息

推送JSON示例

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

```
08:14:03.564Z"}}
```

Json字段说明

参数名	类型	说明
assetId	String	设备号
commandType	String	指令类型
content	String	指令返回内容
dateTime	String	时间（UTC时间）

5.从机数据

从机数据包括从机设备的定位数据，包括主锁/设备号，主锁/从机定位时间，经纬度，速度，里程等信息

JSON字符串示例

```
{"assetId":"8012600005","longitude":113.92294,"latitude":22.670017,"speed":0,"direction":0,"gpsTime":"2023-12-07T08:30:31Z","recvTime":"2023-12-07T08:30:52.474Z","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}
```

Json字段说明

参数名	类型	说明
assetId	String	主锁设备号
longitude	Double	经度(WGS-84)
latitude	Double	纬度(WGS-84)
speed	Integer	速度(km/h)
direction	Integer	方向(0~360)
mileage	Long	里程(km)

gpsTime	String	主锁定位时间（UTC时间）
recvTime	String	主锁接收时间（UTC时间）
subAssetID	String	从机设备号
subGpsTime	String	从机定位时间（UTC时间）
subGpsTimestamp	long	从机定位时间时间戳
battery	Integer	从机电量，255表示充电中
voltage	String	从机电压 (V)
sensorType	Integer	从机类型 1-JT126 , 4-JT709 , 5-JT801 , 6-JT802
locStatus	Integer	从机锁状态 0:关 1:开
locRope	Integer	从机锁绳状态 0:插入 1: 拔出
rsssi	Integer	RSSI
humidity	Integer	从机湿度 0表示无
temperature	double	从机温度 -1000表示无
locTimes	Integer	从机开锁次数
eventType	Integer	从机事件 -1:无 0:关锁事件 1:蓝牙开锁事件 2:开后盖报警 3:远程开锁事件 4:锁绳剪断报警 5:按键唤醒事件 6:心跳包事件 7: 充电唤醒事件 8/20:拔出锁绳事件 9: RFID开锁事件 10: 刷非法卡报警 14: 从机信号丢失报警 15: 阀门关闭事件 16: 阀门打开事件 17: 低电量报警 18: 防拆卸报警 19: 电子仓拆卸事件 21: 锁绳插入 22: 蓝牙连接唤醒 23: 应急仓打开报警 24: 应急仓关闭报警 25: 阀门异常打开报警 26: 锁销关闭事件 27: 锁销开启事件 28:关锁异常 29:电机异常 30:NFC触发
statusJson	String	针对802设备的状态数据（Json字符串），其它类型设备数据可忽略（可参考下面的状态JSON说明）

statusJson字段说明

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

--	--	--

参数名	类型	说明
bottomDisassembly	Integer	结构防拆卸状态,0:无拆卸, 1拆卸
emergencyKey	Integer	应急钥匙状态 0: 封存, 1: 启用
lockMotor	电机状态: 0: 电机关, 1: 电机开	
structuralDisassembly	Integer	结构防拆卸状态,0:无拆卸, 1拆卸
lockValve	Integer	阀门状态,0:关闭, 1打开
lockKnob	Integer	旋钮状态, 0: 关闭, 1: 打开
lockRope	Integer	锁绳状态 0: 插入; 1: 拔出
gateway	Integer	无意义, 可忽略

推送日志报表

用户配置好推送规则后, 可以在推送日志中查询平台主动推送产生的日志, 方便后续查看推送记录, 对异常问题进行分析

序号	公司名称	设备ID	规则名称	数据类型	时间	客户端地址	推送内容	推送结果	响应时长
1		79420200001		定位数据	2022-08-19 16:20:33	http://120.25.245.20:3485/...	{"assetId":"794202000001","b...	成功	6毫秒
2		868012030074		定位数据	2022-08-19 16:20:32	http://120.25.245.20:3485/...	{"assetId":"868012030074","b...	成功	6毫秒
3		8062380001		定位数据	2022-08-19 16:20:32	http://120.25.245.20:3485/...	{"assetId":"8062380001","batt...	成功	7毫秒
4		8052400001		定位数据	2022-08-19 16:20:32	http://120.25.245.20:3485/...	{"assetId":"8052400001","batt...	成功	6毫秒
5		796206001103		定位数据	2022-08-19 16:20:31	http://120.25.245.20:3485/...	{"assetId":"796206001103","b...	成功	6毫秒
6		8062380001		指令应答	2022-08-19 16:20:31	http://120.25.245.20:3485/...	{"assetId":"8062380001","com...	成功	6毫秒
7		8002400004		定位数据	2022-08-19 16:20:30	http://120.25.245.20:3485/...	{"assetId":"8002400004","batt...	成功	7毫秒
8		794202000002		定位数据	2022-08-19 16:20:30	http://120.25.245.20:3485/...	{"assetId":"794202000002","b...	成功	6毫秒
9		794202000011		定位数据	2022-08-19 16:20:30	http://120.25.245.20:3485/...	{"assetId":"794202000011","b...	成功	6毫秒
10		876011000026		定位数据	2022-08-19 16:20:30	http://120.25.245.20:3485/...	{"assetId":"876011000026","b...	成功	6毫秒
11		8000631491		定位数据	2022-08-19 16:20:28	http://120.25.245.20:3485/...	{"assetId":"8000631491","batt...	成功	6毫秒
12		8150640002		定位数据	2022-08-19 16:20:28	http://120.25.245.20:3485/...	{"assetId":"8150640002","batt...	成功	6毫秒

Webhook客户端代码示例

[C#\(.Net\)代码示例](#)

[Java代码示例](#)

[Python代码示例](#)

C#(.Net)

示例说明

针对一些报警以及事件类型ID，可参考 [报警自定义类型说明](#)

有些字段数据可能会因为不同设备类型的原因，字段数据无意义，如果用不上，可忽略

一、定位数据

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

TerminalLocation对象详解

```
/// <summary>
/// 定位数据
/// </summary>
public class TerminalLocation
{
    /// <summary>
    /// 设备号
    /// </summary>
    public string assetId { get; set; }
    /// <summary>
    /// 经度
    /// </summary>
    public double longitude { get; set; } = 0.0;
    /// <summary>
    /// 纬度
    /// </summary>
    public double latitude { get; set; } = 0.0;
    /// <summary>
    /// 速度
    /// </summary>
    public int speed { get; set; }
    /// <summary>
```

```

/// 方向
/// </summary>
public int direction { get; set; } = 0;
/// <summary>
/// 里程
/// </summary>
public long mileage { get; set; } = 0L;
/// <summary>
/// 定位时间 ( UTC时间 )
/// </summary>
public string gpsTime { get; set; }
/// <summary>
/// 接收时间 ( UTC时间 )
/// </summary>
public string recvTime { get; set; }
/// <summary>
/// 定位类型 ( 0 : 不定位 ; 1 : GPS定位 ; 2 : 基站定位 )
/// </summary>
public int locType { get; set; } = 0;
/// <summary>
/// GPRS信号
/// </summary>
public int cellSignal { get; set; } = 0;
/// <summary>
/// 卫星信号
/// </summary>
public int gnssSignal { get; set; } = 0;
/// <summary>
/// 小区码数据 , MNC , MCC , LAC , CID
/// </summary>
public string cells { get; set; }
/// <summary>
/// 电量
/// </summary>
public int battery { get; set; }
/// <summary>
/// 电压
/// </summary>
public string voltage { get; set; }
/**
 * 锁/车载状态JSON
 */
public string statusJson { get; set; }
/**
 * 拓展信息JSON
 */

```

```
public string expandInfo { get; set; }
}
```

锁/车载状态JSON说明

```
/// <summary>
/// 锁/车载状态JSON
/// </summary>
public class StatusJson
{
    /// 锁状态信息JSON说明 (针对电子锁设备)
    /// <summary>
    /// 锁绳状态(1: 拔出 0:插入 -1:无)
    /// </summary>
    public int lockRope { get; set; }
    /// <summary>
    /// 锁状态 (0:关 1:开)
    /// </summary>
    public int lockStatus { get; set; }

    /// 车载状态信息JSON说明 (针对GP系列, 部标设备等车载设备)
    /// <summary>
    /// 引擎开关状态 ( 1:开 0:关 -1:无)
    /// </summary>
    public int acc { get; set; }
    /// <summary>
    /// 断油电开关状态 (1:开 0:关 -1:无)
    /// </summary>
    public int fuelCut { get; set; }
    /// <summary>
    /// 开关门状态 (1:开 0:关 -1:无)
    /// </summary>
    public int door { get; set; }
    /// <summary>
    /// 发动机状态 (1:开 0:关 -1:无)
    /// </summary>
    public int engine { get; set; }
}
```

拓展信息JSON说明

```
/// <summary>
/// 扩展字段
/// </summary>
public class ExpandInfo
```

```

{
    /// <summary>
    /// 温度 -1000表示无
    /// </summary>
    public int temperature { get; set; }
    /// <summary>
    /// 湿度 0表示无
    /// </summary>
    public int humidity { get; set; }
    /// <summary>
    /// 油位值-1表示无 ( "-1 , -1 , -1" )
    /// </summary>
    public int fuels { get; set; }
    /// <summary>
    /// 加速度 ( 格式 : "x:1;y:-29;z:-2903" )
    /// </summary>
    public string acceleration { get; set; }
    /// <summary>
    /// 光照度 ( Lux )
    /// </summary>
    public float lux { get; set; }
    /// <summary>
    /// 气压(pa)
    /// </summary>
    public float pressure { get; set; }
    /// <summary>
    /// 姿态 ( 格式 : "x:1;y:-29;z:-2903" )
    /// </summary>
    public string fPosture { get; set; }
    /// <summary>
    /// 备用电池 "55,3.88,0"
    /// </summary>
    public string backBattery { get; set; }
    /// <summary>
    ///数据类型 0 : 实时 ; 1:补报 ; 2 : 报警
    /// </summary>
    public int reportType { get; set; }
    /// <summary>
    /// 网络类型 0 : 未知 1 : 1G 2 : 2G 3:3G 4:4G 5:5G 其它不显示
    /// </summary>
    public int networkType { get; set; }
}

```

二、报警数据

```

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

```

TerminalAlarm对象详解

```

/// <summary>
/// 报警数据
/// </summary>
public class TerminalAlarm
{
    /// <summary>
    /// 设备号
    /// </summary>
    public string assetId { get; set; }
    /// <summary>
    /// 报警类型
    /// </summary>
    public int alarmType { get; set; }
    /// <summary>
    /// UTC时间
    /// </summary>
    public string dateTime { get; set; }
    /// <summary>
    /// 经度(WGS-84)
    /// </summary>
    public double longitude { get; set; } = 0.0;
    /// <summary>
    /// 纬度(WGS-84)
    /// </summary>
    public double latitude { get; set; } = 0.0;
    /// <summary>
    /// 速度(km/h)
    /// </summary>
    public int speed { get; set; }
    /// <summary>
    /// 里程(km)
    /// </summary>
    public long mileage { get; set; }
}

```

```

    /// <summary>
    /// 小区码(mcc,mnc,lac,cellid)
    /// </summary>
    public string cells { get; set; }
    /// <summary>
    /// 报警描述
    /// </summary>
    public string describe { get; set; }
    /// <summary>
    /// 附件ID
    /// </summary>
    public string fileIndex{ get; set; }
}

```

三、事件数据

```

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

```

TerminalEvent对象详解

```

    /// <summary>
    /// 事件数据
    /// </summary>
    public class TerminalEvent
    {
        /// <summary>
        /// 设备号
        /// </summary>
        public string assetId { get; set; }
        /// <summary>
        /// 事件类型 1：开锁 2：关锁 3：出区域施封 4：进区域解封 5：开箱/区 6：
        关箱/门
        /// </summary>
        public int eventType { get; set; }
        /// <summary>
        /// 开锁类型 1:刷卡开锁 2：远程开锁 3：蓝牙开锁 4：剪绳开锁 5：短信开锁

```

6 : 区域触发

```

    /// </summary>
    public int unLockType { get; set; }
    /// <summary>
    /// UTC时间
    /// </summary>
    public string dateTime { get; set; }
    /// <summary>
    /// 经度(WGS-84)
    /// </summary>
    public double longitude { get; set; } = 0.0;
    /// <summary>
    /// 纬度(WGS-84)
    /// </summary>
    public double latitude { get; set; } = 0.0;
    /// <summary>
    /// 速度(km/h)
    /// </summary>
    public int speed { get; set; }
    /// <summary>
    /// 里程(km)
    /// </summary>
    public long mileage { get; set; }
    /// <summary>
    /// 小区码数据(mcc,mnc,lac,cellid)
    /// </summary>
    public string cells { get; set; }
    /// <summary>
    /// 刷卡开锁卡号
    /// </summary>
    public string card { get; set; }
    /// <summary>
    /// 开锁密码
    /// </summary>
    public string password { get; set; }
}

```

四、指令应答数据

```

[HttpPost]
public MBackResult ReceiveInsData([FromBody]TerminalCommand parm)
{
    Log.Instance.Warn("指令数据：" + JsonConvert.SerializeObject(parm));
    MBackResult result = new MBackResult();
}

```

```

        result.Result = 200;
        return result;
    }

```

TerminalCommand对象详解

```

    /// <summary>
    /// 指令应答数据
    /// </summary>
    public class TerminalCommand
    {
        /// <summary>
        /// 设备号
        /// </summary>
        public string assetId { get; set; }
        /// <summary>
        /// 指令类型
        /// </summary>
        public string commandType { get; set; }
        /// <summary>
        /// 指令内容
        /// </summary>
        public string content { get; set; }
        /// <summary>
        /// UTC时间
        /// </summary>
        public string dateTime { get; set; }
    }

```

五、从机数据

```

    [HttpPost]
    public MBackResult ReceiveInsData([FromBody]SlaveMachineLocation pa
rm)
    {
        Log.Instance.Warn("从机数据：" + JsonConvert.SerializeObject(par
m));
        MBackResult result = new MBackResult();
        result.Result = 200;
        return result;
    }

```

SlaveMachineLocation对象详解

```
/// <summary>
/// 从机数据
/// </summary>
public class SlaveMachineLocation
{
    /// <summary>
    /// 主锁设备ID
    /// </summary>
    public string assetId { get; set; }
    /// <summary>
    /// 主锁经度(WGS-84)
    /// </summary>
    public double longitude { get; set; } = 0.0;
    /// <summary>
    /// 主锁纬度(WGS-84)
    /// </summary>
    public double latitude { get; set; } = 0.0;
    /// <summary>
    /// 主锁速度
    /// </summary>
    public int speed { get; set; }
    /// <summary>
    /// 主锁方向
    /// </summary>
    public int direction { get; set; }
    /// <summary>
    /// 主锁定位时间 ( UTC时间 )
    /// </summary>
    public string gpsTime { get; set; }
    /// <summary>
    /// 主锁接收时间 ( UTC时间 )
    /// </summary>
    public string recvTime { get; set; }
    /// <summary>
    /// 从机定位时间 ( UTC时间 )
    /// </summary>
    public string subGpsTime { get; set; }
    /// <summary>
    /// 从机电量
    /// </summary>
    public int battery { get; set; }
    /// <summary>
    /// 从机电压
    /// </summary>
    public string voltage { get; set; }
    /// <summary>
```

```

    /// 从机设备号
    /// </summary>
    public string subAssetID { get; set; }
    /// <summary>
    /// 从机类型 1-JT126 4-JT709 5-JT801 6-JT802
    /// </summary>
    public int sensorType { get; set; }
    /// <summary>
    /// 针对802设备的状态数据 (Json字符串), 其它类型设备数据可忽略
    /// </summary>
    public string statusJson { get; set; }
    /// <summary>
    /// 从机锁状态 0-关 1-开
    /// </summary>
    public int locStatus { get; set; }
    /// <summary>
    /// 从机锁绳状态 0-关 1-开
    /// </summary>
    public int locRope { get; set; }
    /// <summary>
    ///RSSI
    /// </summary>
    public int rssi { get; set; }
    /// <summary>
    /// 从机温度 -1000表示无
    /// </summary>
    public double temperature { get; set; } = -1000.0;
    /// <summary>
    /// 从机湿度 0表示无
    /// </summary>
    public int humidity { get; set; }
    /// <summary>
    /// 从机事件 -1:无 0:关锁事件 1:蓝牙开锁事件 2:开后盖报警 3:远程开锁事件
    4:锁绳剪断报警 5:按键唤醒事件 6:心跳包事件 7:充电唤醒事件 8/20:拔出锁绳事件 9
    :RFID开锁事件 10:刷非法卡报警 14:从机信号丢失报警 15:阀门关闭事件 16:阀门
    打开事件 17:低电量报警 18:防拆卸报警 19:电子仓拆卸事件 21:锁绳插入 22:蓝牙连
    接唤醒 23:应急仓打开报警 24:应急仓关闭报警 25:阀门异常打开报警 26:锁销关闭事
    件 27:锁销开启事件 28:关锁异常 29:电机异常 30:NFC触发
    /// </summary>
    public int eventType { get; set; } = -1;
    /// <summary>
    /// 从机开锁次数
    /// </summary>
    public int locTimes { get; set; }
    /// <summary>
    /// 从机定位时间戳

```

```
/// </summary>
```

```
public long subGpsTimestamp { get; set; }
```

```
}
```



Java

示例说明

针对一些报警以及事件类型ID，可参考 [自定义类型说明](#)

有些字段数据可能会因为不同设备类型的原因，字段数据无意义，如果用不上，可忽略

一、定位数据

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

TerminalLocation对象详解

```
@Data
public class TerminalLocation {
    /**
     * 设备号
     */
    public String assetId;
    /**
     * 经度 (WGS-84)
     */
    public Double longitude = 0.0d;
    /**
     * 纬度 (WGS-84)
     */
    public Double latitude = 0.0d;
    /**
     * 速度(km/h)
     */
    public Integer speed;
    /**
     * 方向(0~360)
     */
}
```

```
public Integer direction = 0;
/**
 * 里程(km)
 */
public Long mileage = 0L;
/**
 * 定位时间(UTC)
 */
public String gpsTime;
/**
 * 接收时间(UTC)
 */
public String recvTime;
/**
 * 定位类型
 */
public Integer locType = 0;
/**
 * GPRS信号
 */
public Integer cellSignal = 0;
/**
 * 卫星信号
 */
public Integer gnssSignal = 0;
/**
 * 小区码数据 , MNC , MCC, LAC, CID
 */
public String cells;
/**
 * 电量
 */
public Integer battery;
/**
 * 电压
 */
public String voltage;
/**
 * 车载/锁状态JSON
 */
public String statusJson;
/**
 * 拓展信息JSON
 */
public String expandInfo;
}
```

车载/锁信息JSON说明

```

@Data
public class StatusJson {
    /**
     * 锁状态 (0:关 1:开)
     */
    private Integer lockStatus;
    /**
     * 锁绳状态(1: 拔出 0:插入 -1:无)
     */
    private Integer lockRope;
    /**
     * 索条状态 (0:未知; 1:在线; 2:弯曲; 3:锯断; 4:丢失)
     */
    private Integer lockBar;
    /**
     * 引擎开关状态 ( 1:开 0:关 -1:无)
     */
    private Integer acc;
    /**
     * 断油电开关状态 (1:开 0:关 -1:无)
     */
    private Integer fuelCut;
    /**
     * 开关门状态 (1:开 0:关 -1:无)
     */
    private Integer door;
    /**
     * 发动机状态 (1:开 0:关 -1:无)
     */
    private Integer engine;
}
}

```

拓展信息JSON说明

```

@Data
public class ExpendInfo {
    /**
     * 温度 -1000表示无
     */
    private String temperature;
    /**
     * 湿度 0表示无
     */
}

```

```

    */
    private String humidity;
    /**
     * 备用电池 ("55,3.88,0")
     */
    private String backBattery;
    /**
     * 油位值; -1: 标识无效
     */
    private String fuels;
    /**
     * 加速度(格式如: "x:1;y:-29;z:-2903")
     */
    private String acceleration;
    /**
     * 光照度 (Lux)
     */
    private String lux;
    /**
     * 气压 (pa)
     */
    private String pressure;
    /**
     * 姿态信息(格式如: "x:1;y:-29;z:-2903")
     */
    private String posture;
    /**
     * 0:未知;1:1G;2:2G;3:3G;4:4G;5:5G
     */
    private String networkType;

    /**
     * 0:实时;1:补报;2:报警
     */
    private String reportType;
}

```

二、报警数据

```

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

```

```
        return result;
    }
```

TerminalAlarm对象详解

```
@Data
public class TerminalAlarm {
    /**
     * 设备号
     */
    public String assetId;
    /**
     * 报警类型
     */
    public int alarmType;
    /**
     * 时间 ( UTC )
     */
    public String dateTime;
    /**
     * 经度 ( WGS-84 )
     */
    public Double longitude = 0.0d;
    /**
     * 纬度 ( WGS-84 )
     */
    public Double latitude = 0.0d;
    /**
     * 速度(km/h)
     */
    public Integer speed;
    /**
     * 里程(km)
     */
    public Long mileage;
    /**
     * 小区码(mcc,mnc,lac,cellid)
     */
    public String cells;
    /**
     * 报警描述
     */
    public String describe;
    /**
     * 附件ID
     */
}
```

```

    */
    public String fileIndex;
}

```

三、事件数据

```

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

```

TerminalEvent对象详解

```

@Data
public class TerminalEvent {
    /**
     * 设备号
     */
    public String assetId;
    /**
     * 事件类型
     */
    public Integer eventType;
    /**
     * 开锁类型
     */
    public Integer unLockType;
    /**
     * 时间 ( utc )
     */
    public String dateTime;
    /**
     * 经度(wgs-84)
     */
    public Double longitude = 0.0d;
    /**
     * 纬度(wgs-84)
     */
    public Double latitude = 0.0d;
}

```

```

    * 速度(km/h)
    */
    public Integer speed;
    /**
    * 里程(km)
    */
    public Long mileage;
    /**
    * 小区码数据(mcc,mnc,lac,cellid)
    */
    public String cells;
    /**
    * 刷卡开锁卡号
    */
    public String card;
    /**
    * 开锁密码
    */
    public String password;
}

```

四、指令应答数据

```

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

```

TerminalCommand对象详解

```

@Data
public class TerminalCommand {
    /**
    * 设备号
    */
    public String assetId;
    /**
    * 指令类型
    */
    public String commandType;
}

```

```

/**
 * 指令内容
 */
public String content;
/**
 * 时间 ( utc )
 */
public String dateTime;
}

```

五、从机数据

```

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

```

SlaveMachineLocation对象详解

```

/// <summary>
/// 从机数据
/// </summary>
public class SlaveMachineLocation
{
    /// <summary>
    /// 主锁设备ID
    /// </summary>
    public String assetId ;
    /// <summary>
    /// 主锁经度(WGS-84)
    /// </summary>
    public Double longitude = 0.0d;
    /// <summary>
    /// 主锁纬度(WGS-84)
    /// </summary>
    public Double latitude = 0.0d;
    /// <summary>
    /// 主锁速度
    /// </summary>
}

```

```
public Integer speed ;  
/// <summary>  
/// 主锁方向  
/// </summary>  
public Integer direction ;  
/// <summary>  
/// 主锁定位时间 ( UTC时间 )  
/// </summary>  
public String gpsTime ;  
/// <summary>  
/// 主锁接收时间 ( UTC时间 )  
/// </summary>  
public String recvTime ;  
/// <summary>  
/// 从机定位时间 ( UTC时间 )  
/// </summary>  
public String subGpsTime ;  
/// <summary>  
/// 从机电量  
/// </summary>  
public Integer battery ;  
/// <summary>  
/// 从机电压  
/// </summary>  
public String voltage ;  
/// <summary>  
/// 从机设备号  
/// </summary>  
public String subAssetID ;  
/// <summary>  
/// 从机类型 1-JT126 4-JT709 5-JT801 6-JT802  
/// </summary>  
public Integer sensorType ;  
/// <summary>  
/// 针对802设备的状态数据 ( Json字符串 ) , 其它类型设备数据可忽略  
/// </summary>  
public String statusJson ;  
/// <summary>  
/// 从机锁状态 0-关 1-开  
/// </summary>  
public Integer locStatus ;  
/// <summary>  
/// 从机锁绳状态 0-关 1-开  
/// </summary>  
public Integer locRope;  
/// <summary>
```

```
    ///RSSI
    /// </summary>
    public Integer rssi ;
    /// <summary>
    /// 从机温度 -1000表示无
    /// </summary>
    public Double temperature = -1000.0;
    /// <summary>
    /// 从机湿度 0表示无
    /// </summary>
    public Integer humidity ;
    /// <summary>
    /// 从机事件 -1:无 0:关锁事件 1:蓝牙开锁事件 2:开后盖报警 3:远程开锁事
    件 4:锁绳剪断报警 5:按键唤醒事件 6:心跳包事件 7:充电唤醒事件 8/20:拔出锁绳事件 9
    :RFID开锁事件 10:刷非法卡报警 14:从机信号丢失报警 15:阀门关闭事件 16:阀门
    打开事件 17:低电量报警 18:防拆卸报警 19:电子仓拆卸事件 21:锁绳插入 22:蓝牙连
    接唤醒 23:应急仓打开报警 24:应急仓关闭报警 25:阀门异常打开报警 26:锁销关闭事
    件 27:锁销开启事件 28:关锁异常 29:电机异常 30:NFC触发
    /// </summary>
    public Integer eventType = -1;
    /// <summary>
    /// 从机开锁次数
    /// </summary>
    public Integer locTimes ;
    /// <summary>
    /// 从机定位时间戳
    /// </summary>
    public Long subGpsTimestamp ;
}
```

Webhook Server-Python

Python code简单示例:

```
# 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规则配置说明

概述

1. Jointech提供的MQTT推送服务，是一种将设备数据实时发送给数据使用方的服务。当设备上传数据后，推送服务会第一时间感知，在将数据整理打包后，发送至MQTT服务器，客户数据接收服务从MQTT服务器获取数据，然后客户根据自身业务自行处理数据。
2. MQTT规则配置界面，主要是方便用户配置哪些设备进行MQTT推送服务
3. 订阅主题说明

MQTT规则配置

用户可以在此界面中，配置需要进行MQTT推送的终端，通过选择数据源，可以进行多方式的选择所需要推送的终端。



数据源说明

1. 公司全部设备

表示当前公司及其子公司下的所有终端

2. 公司直属设备

表示当前公司及下的所有终端

3. 公司指定设备

表示当前公司及其子公司下的指定终端

订阅主题及负载说明

1.服务器地址及说明

参数名	说明
服务器地址	mqtt://mqtt.assetscontrols.com:1883
协议版本	MQTT3.1.1
QoS等级	建议QoS0（最多一次）

2.客户端账户及密码

为当前MQTT规则配置的登录账户及密码（MD5加密后），也可直接在该规则的详情界面中查看

。

3.订阅说明

将{accessKey}替换为系统分配的实际值，{deviceNum}替换为设备的ID，{subLockNum}替换为设备子锁的ID

订阅主题	说明
upload/{accessKey}/#	订阅全部设备所有类型数据
upload/{accessKey}/{deviceNum}/#	订阅某一个设备所有类型数据
upload/{accessKey}/+/location	订阅全部设备定位数据
upload/{accessKey}/{deviceNum}/location	订阅某一个设备定位数据
upload/{accessKey}/+/alarm	订阅全部设备报警数据
upload/{accessKey}/{deviceNum}/alarm	订阅某一个设备报警数据
upload/{accessKey}/+/event	订阅全部设备事件数据
upload/{accessKey}/{deviceNum}/event	订阅某一个设备事件数据
upload/{accessKey}/+/cmd	订阅全部设备指令应答数据

upload/{accessKey}/{deviceNum}/cmd	订阅某一个设备指令应答数据
upload/{accessKey}/{deviceNum}/{subLockNum}/sublock	订阅某一个设备某一个从机状态数据
upload/{accessKey}/{deviceNum}/+/sublock	订阅某一个设备所有从机状态数据
download/{accessKey}/{deviceNum}/cmd	下发某一个设备指令到终端数据

3.1 终端状态上报

主题: upload/{accessKey}/{deviceNum}/location

描述: 终端的定位状态数据

示例:

```
{
  "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",
    "networkType": "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,
  "recvTime": "2025-02-07T07:25:20.832Z",
  "speed": 0,
  "statusJson": {
    "lockRope": 0,
    "lockStatus": 0
  },
  "voltage": "0.0"
}
```

示例Json字符串说明:

参数名	类型	说明
assetId	String	设备号
longitude	Double	经度(WGS-84)
latitude	Double	纬度(WGS-84)
speed	Integer	速度(km/h)
direction	Integer	方向(0~360)
mileage	Long	里程(km)
gpsTime	String	定位时间 (UTC时间)
recvTime	String	接收时间 (UTC时间)
locType	Integer	定位类型 (0: 不定位; 1: GPS定位; 2: 基站定位)

cellSignal	Integer	GPRS信号
gnssSignal	Integer	卫星信号
cells	String	小区码数据，以 MNC, MCC,LAC,CID的格式显示
battery	Integer	电量，255表示充电中
voltage	String	电压 (V)
statusJson	String	锁/车载状态JSON（可参考下面的状态JSON说明）
expandInfo	String	拓展信息JSON（可参考下面的扩展信息JSON说明）

锁状态JSON说明

参数名	类型	说明
lockRope	Integer	锁绳状态(1: 拔出 0:插入 -1:无)
lockStatus	Integer	锁状态 (0: 关 1: 开)

车载状态JSON说明

参数名	类型	说明
acc	Integer	引擎开关状态 (1: 开 0: 关 -1: 无)
fuelCut	Integer	断油电开关状态 (1: 开 0: 关 -1: 无)
door	Integer	开关门状态 (1: 开 0: 关 -1: 无)
engine	Integer	发动机状态 (1: 开 0: 关 -1: 无)

拓展信息JSON说明

参数名	类型	说明
temperature	String	温度 -1000表示无
humidity	String	湿度 0表示无
fuels	String	油位值 -1表示无 (“-1, -1, -1”)
fAcceleration	String	加速度(“x:1;y:-29;z:-2903”)

lux	float	光照度
pressure	float	气压(pa)
posture	String	姿态 (“x:1;y:-29;z:-2903”)
fVoltage	Double	电压值
backBattery	String	备用电池 (“55,3.88,0”)
fReportType	Integer	数据类型 (0: 实时; 1:补报; 2: 报警)
fNetworkType	Integer	网络类型 (0: 未知 1: 1G 2: 2G 3:3G 4:4G 5:5G)

3.2 终端报警上报

主题: upload/{accessKey}/{deviceNum}/alarm

描述: 终端的报警数据, 如低电量, 撞击报警

示例:

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

示例Json字符串说明:

参数名	类型	说明
assetId	String	设备号
alarmType	Integer	报警类型 (可参考下面的报警类型说明)
dateTime	String	时间 (UTC时间)
longitude	Double	经度(WGS-84)
latitude	Double	纬度(WGS-84)
speed	Integer	速度(km/h)
mileage	Long	里程(km)
cells	String	小区码数据, 以 MNC, MCC,LAC,CID的格式显示
describe	String	描述

fileIndex	String	附件流水号
-----------	--------	-------

报警类型说明：

报警类型	报警名称
1	超速报警
2	疲劳驾驶
3	危险预警
4	GNSS模块发生故障
5	GNSS天线未接或被剪断
6	GNSS天线短路
7	终端主电源欠压
8	终端主电源掉电
9	终端LCD或显示器故障
10	TTS模块故障
11	摄像头故障
12	道路运输证IC卡模块故障
13	超速预警
15	断电报警
19	超时停车
23	路线偏离报警
24	车辆VSS故障
26	车辆被盗
27	车辆非法点火
28	车辆非法位移
29	碰撞预警

30	侧翻预警
31	非法开门报警
32	视频信号丢失报警
33	视频信号遮挡报警
34	存储单元故障报警
35	其他视频设备故障报警
36	客车超员报警
37	异常驾驶行为报警
38	特殊报警录像达到存储阈值报警
40	锁绳剪断
41	震动
42	长时间开锁
43	开锁密码连续5次错误
44	刷非法卡
45	低电量
46	开后盖
47	卡锁
48	进区域报警
49	出区域报警
50	启用后备电池
51	SOS
52	拖吊报警
54	油位报警
55	进热点报警

56	出热点报警
57	进道路报警
58	出道路报警
63	温湿度报警
64	前向碰撞报警
65	车道偏离报警
66	车距过近报警
67	行人碰撞报警
68	频繁变道报警
69	道路标识超限报警
70	障碍物报警
71	道路标志识别事件
72	主动抓拍事件
73	实线变道报警
74	车厢过道行人监测报警
92	疲劳驾驶报警
93	接打电话报警
94	抽烟报警
95	分神驾驶报警
96	驾驶员异常报警
97	自动抓拍事件
98	驾驶员变更事件
99	探头遮挡报警
100	超时驾驶报警

101	未系安全带报警
102	红外阻断型墨镜失效报警
103	双脱把报警
104	玩手机报警
110	胎压报警
125	后方接近报警
126	左侧后方接近报警
127	右侧后方接近报警
131	急加速报警
132	急减速报警
133	急转弯报警
134	怠速报警
135	异常熄火报警
136	空挡滑行报警
137	发动机超转报警
141	超速报警
145	超过车辆额定载重报警
146	超过道路承重报警
151	限高报警
160	安全带报警
161	紧急刹车报警
162	空挡滑行
163	GPRS重连
164	调度屏连接

165	调度屏断开
166	CANBUS断开连接报警
167	CANBUS故障码上传报警
168	限制开车报警
180	主机拆卸
190	上盖破坏
191	Gps天线干扰
192	低电量休眠报警
193	锁异常
194	开锁密码错误
195	没有定位不执行开锁
196	围栏外禁止开锁
197	姿态报警
198	从机信号丢失
199	监管状态禁止开锁
200	主电池更换
201	自检异常
202	存储空间不足
203	锁条弯曲
204	锁条锯断
205	锁条丢失
206	施封状态禁止开锁
316	光感报警
332	气压报警

400	超时怠速报警
402	围栏超时停留报警
403	出围栏未上锁报警
404	晚发预警
405	晚发报警
406	晚到预警
407	晚到报警

3.3 终端事件上报

主题: upload/{accessKey}/{deviceNum}/event

描述: 终端的事件数据, 如开锁, 关锁

示例:

```
{"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}
```

示例Json字符串说明:

事件类型	事件名称
1	开锁
2	关锁
3	出区域施封
4	进区域解封
5	开箱/门
6	关箱/门
7	蓝牙施封
8	蓝牙解封

9	远程施封
10	远程解封
11	抓拍
12	定时拍
13	离开起点
14	到达目的地
15	离开目的地
16	完成运单
17	进入途经点
18	离开途经点
19	拔出锁绳/按下开锁按钮(JT705A/JT705C)

3.4 指令应答上报

主题: `upload/{accessKey}/{deviceNum}/cmd`

描述: 终端响应服务器下发的指令

指令应答说明: 不同的指令应答结果不一样, 具体指令及其说明, 请参考协议文档

示例:

//正常回复

```
{
  "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"}

```

//失败回复 (Device not online)

```
{
  "assetId": "8294630003",
  "commandType": "MQTT_CMD",
  "content": "Device not online!",
  "dateTime": "2025-01-07T09:49:43.158Z"}

```

//失败回复 (Device has not been registered)

```
{
  "assetId": "8294630481",
  "commandType": "MQTT_CMD",
  "content": "Device has not been registered!",
  "dateTime": "2025-01-07T09:48:32.975Z"}

```

示例Json字符串说明:

可参考数据引擎服务说明中: 数据类型说明-> 指令应答数据

3.5 从机状态上报

主题: upload/{accessKey}/{deviceNum}/{subLockNum}/sublock

描述: 从机的定位状态数据

示例:

```
{
  "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",
  "rsi": 15,
  "sensorType": 4,
  "speed": 0,
  "statusJson": "{\"lockRope\":0,\"gateway\":0}",
  "subAssetID": "E0171E0366",
  "subGpsTime": "2025-02-07T07:15:14Z",
  "subGpsTimestamp": 1738912514000,
  "temperature": -1000.0,
  "voltage": "3.75"
}
```

示例Json字符串说明:

参数名	类型	说明
assetId	String	主锁设备号
longitude	Double	经度(WGS-84)
latitude	Double	纬度(WGS-84)
speed	Integer	速度(km/h)
direction	Integer	方向(0~360)
mileage	Long	里程(km)
gpsTime	String	主锁定位时间 (UTC时间)
recvTime	String	主锁接收时间 (UTC时间)
subAssetID	String	从机设备号
subGpsTime	String	从机定位时间 (UTC时间)
subGpsTimestamp	long	从机定位时间时间戳
battery	Integer	从机电量, 255表示充电中
voltage	String	从机电压 (V)
sensorType	Integer	从机类型 1-JT126, 4-JT709, 5-JT801, 6-JT802
locStatus	Integer	从机锁状态 0:关 1:开
locRope	Integer	从机锁绳状态 0:插入 1:拔出

rsi	Integer	RSSI
humidity	Integer	从机湿度 0表示无
temperature	double	从机温度 -1000表示无
locTimes	Integer	从机开锁次数
eventType	Integer	从机事件 -1:无 0:关锁事件 1:蓝牙开锁事件 2:开后盖报警 3:远程开锁事件 4:锁绳剪断报警 5:按键唤醒事件 6:心跳包事件 7: 充电唤醒事件 8/20:拔出锁绳事件 9: RFID开锁事件 10: 刷非法卡报警 14: 从机信号丢失报警 15: 阀门关闭事件 16: 阀门打开事件 17: 低电量报警 18: 防拆卸报警 19: 电子仓拆卸事件 21: 锁绳插入 22: 蓝牙连接唤醒 23: 应急仓打开报警 24: 应急仓关闭报警 25: 阀门异常打开报警 26: 锁销关闭事件 27: 锁销开启事件 28:关锁异常 29:电机异常 30:NFC触发
statusJson	String	针对802设备的状态数据（Json字符串），其它类型设备数据可忽略（可参考下面的状态JSON说明）

statusJson字段说明

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

参数名	类型	说明
bottomDisassembly	Integer	结构防拆卸状态,0:无拆卸, 1拆卸
emergencyKey	Integer	应急钥匙状态 0: 封存, 1: 启用
lockMotor	电机状态: 0: 电机关, 1: 电机开	
structuralDisassembly	Integer	结构防拆卸状态,0:无拆卸, 1拆卸
lockValve	Integer	阀门状态,0:关闭, 1打开
lockKnob	Integer	旋钮状态, 0: 关闭, 1: 打开
lockRope	Integer	锁绳状态 0: 插入; 1: 拔出

gateway	Integer	无意义，可忽略
---------	---------	---------

3.6 服务器指令下发

主题：download/{accessKey}/{deviceNum}/cmd

描述：下发指令到终端的数据

指令说明：不同设备类型有不同的指令功能，具体指令及其说明，请参考协议文档
示例：

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

MQTT客户端代码示例

[C#\(.Net\)代码示例](#)

[Java代码示例](#)

[Python代码示例](#)

C#(.Net)

.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 server.");

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

    // Handle received messages
    client.MqttMsgPublishReceived += Client_MqttMsgPublishReceived;

    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.Message);
    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 closed.");
    TryReconnect();
}

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

```

C#(.Net)

}

Java

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      = "u8vng1";//user
String password  = "xxxxxx";//password(MD5-32)
String topic     = "upload/{accessKey}/{deviceNum}/location";
//String topic = "upload/B176C0FF1DFF41D0126BFF7908C76E17/8041254836/Location"; //Subscribe to the location data of this device
//String topic = "upload/B176C0FF1DFF41D0126BFF7908C76E17/+/Location"; //Subscribe to the location data of this configuration device
//String topic = "upload/B176C0FF1DFF41D0126BFF7908C76E17/#"; //Subscribe to 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 address
String clientId = "u8vng1_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 clientId, 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 during
```

```

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 Exception {
    //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, making
    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;
            }
        }
    }
}

```

Java

```
    }  
    TimeUnit.SECONDS.sleep(100);  
} catch (MqttException | InterruptedException e1) {  
    e1.printStackTrace();  
}  
System.out.println("client not connect");  
}  
}
```

Python

Python 示例

```
# 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}')
```

```

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...", reconnect_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
mqtt_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(try_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)
```