金桔产品 API 接口规格书

金桔智能（北京）科技有限公司
地址：北京市海淀区北清路 68 号用友软件园中区 7 号楼 325 室
2019 年 7 月 28 日

文档更新说明

SN.	更新内容	更新人	日期
1	定位：增加胸牌低电模式下 beacon 数据报警	Lorin	20190926
2	定位：添加基站心跳和基站启动类型数据		20191009
3	定位：增加低电量提示		20191112
4	iBeacon：增加 iBeacon 推送接口		20200306

目录

第1章 API 接口文档说明

API 接口文档是金桔智能系列产品的全套接口文档，包括室内定位系列产品、iBeacon 产品、蓝牙网关透传数据、联网门锁等，此接口是系统开发对接的说明性文档，阅读者要具备开发基础。

第2章 配置工具

金桔系列产品均采用微信小程序配置，请自行在微信中搜索：**金桔 BOX**
配置工具为开源项目，请自行下载学习。
地址：https://github.com/jinguolock/wxlocation_demo
注意：我方配置工具为微信小程序且更新比较频繁， 若需同步，部署时请与厂商联系确认版本是否为最新版本。

第3章 室内定位接口

3.1. 文档中相关名词解释

定位设备：定位设备是定位胸牌、定位手环、固定资产标签的统称。
定位位标：定位位标也会描述为锚点、iBeacon，分为室内型和室外型。
定位网关：定位网关包含蓝牙定位网关、蓝牙+LoRa 双模定位网关、LoRa 定位网关。
室内定位接口包括三类接口，ACserver 定位引擎接口、ACserver 推送数据接口、网络串口数据接口。
原始数据推送格式：定位网关通过网络端口所发送的原始数据。

3.2. ACserver 定位引擎接口

ACserver 定位引擎接口是采用我司的定位引擎推送的数据接口，此定位引擎为数据的初始处理，若需在前端提供优异的用户体验，需开发者具备足够的算法基础及前端优化基础。若不具备此能力者，请慎用此接口。

3.2.1. 位置服务说明

位置服务是 ACserver 提供给客户的坐标转换服务，通过 beacon 和基站的信号强度和用户

设置固定设备的坐标信息，计算出移动定位设备当前的相对位置。在定位服务里面，用户对 beacon 和基站进行位置坐标标记，这里需要用户配置 7 个字段的信息。

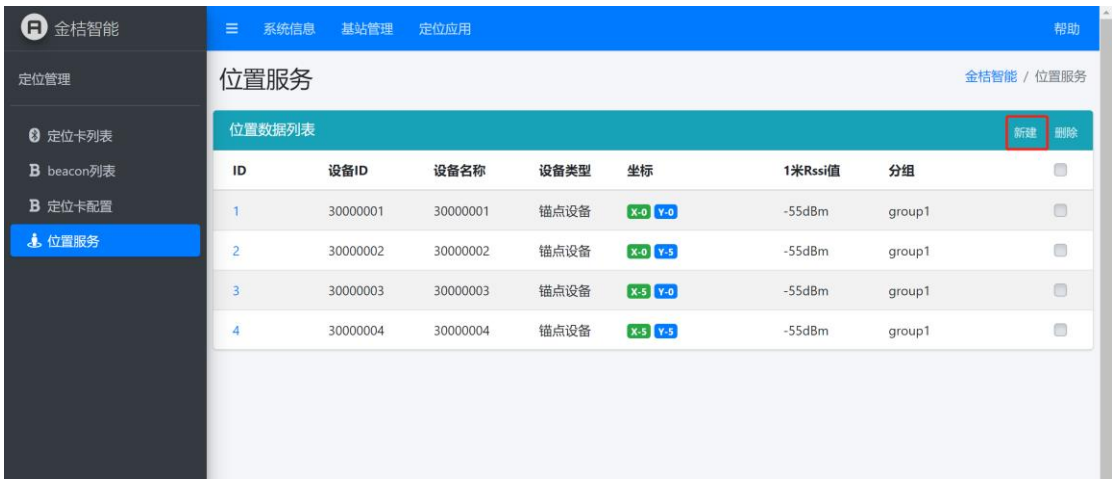
字段	含义
设备 ID	如果添加的是 beacon 设备那么这个 ID 就是 MajorHex+MinorHex，例如：一个 beacon 的 major 是 25517 那么 16 进制是 63AD（注意一定要大写），minor 是 63382，16 进制是 F796，那么这个设备的 id 就是：63ADF796。如果添加的基站设备，那么这个 ID 就是基站的 ID，例如 10000018
设备名称	方便管理，可以任意起名称（最好是字母数字）。
类型	是 BEACON 设备还是基站设备（值为：beacon 或 station）
X 坐标	设置设备的 X 坐标值（float 类型）
Y 坐标	设置设备的 Y 坐标值（float 类型）
1 米 RSSI	这个主要是计算位置时校准需要用到的参数，如果不了解可以使用默认值，当测量位置有误差时可以调整这个参考值 例如：-59
分组名称	这个是对设备的分组，主要作用例如，当胸牌处于 1 楼楼梯口那么在此位置有可能取到 2 楼 beacon 的信号值，对于不在 2 楼的移动的胸牌来说这个设备有可能是增加误差的，所以对设备进行分组管理，那么在 1 楼的胸牌将过滤掉 2 楼的 beacon，有助于精准计算位置。（不能出现汉字）

该服务只有在胸牌搜索到 3 个或以上设备时有效.

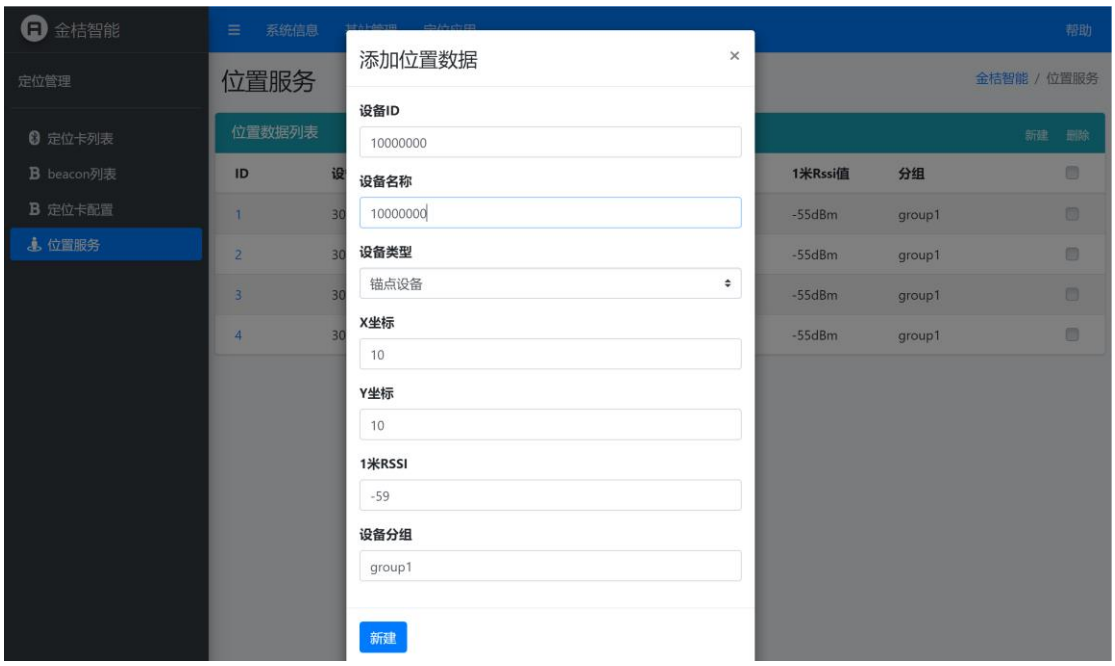
3.2.2. 位置服务手动添加

注意：位置服务示例仅为测试使用，正式使用请使用 API 接口调用。
定位应用——位置服务：

选择新建



添加设备



详细应用请参照说明书。

3.2.3. 位置服务位标增删改查

3.2.3.1. 位标增加

http://127.0.0.1:8888/api/location?cmd=add_pos_device&deviceid=10000004&devicename=10000004&devicetype=beacon&devicex=12&devicey=11&rssi1m=-59&group=ground

参数说明

cmd:命令类型，固定是 add_pos_device

deviceid:要添加设备的 id，如果是基站类型就是基站 id，如果是 beacon 就是 majorHex+minorHex，十六进制合起来

devicename:起个名字，不是汉字都可以
devicetype:如果是基站设备这里填 station 如果是 beacon 就填 beacon
devicex:设备的 x 相对坐标
devicey:设备的 y 相对坐标
rssi1m:1 米处的 rssi 值，一般金桔的 beacon 是-59，这个校准值可以再调节
group:分组的组名，不能是汉字

3.2.3.2.读取当前位标数据

取得所有的 beacon 位置配置
http://127.0.0.1:8888/api/location?cmd=get_pos_devices
该接口将返回一个 json 的数组（array），里面包含 baecon 数据对象（object）
beacon 对象的字段说明
Id: 这个是数据库表 id 是数据唯一标识
DeviceId:如果是基站，就是基站 id，如果是 beacon 就是 majorHex+minorHex，十六进制合起来
DeviceName:设备名，添加的时候配置进去的
DeviceType:如果是基站设备就是 station 如果是 beacon 就是 beacon
Devicex:设备的 x 相对坐标
Devicey:设备的 y 相对坐标
Device1mRssi:1 米处的 rssi 值
DeviceGroup:分组的组名

3.2.3.3.更新位标数据

更新位置数据接口
http://127.0.0.1:8888/api/location?cmd=update_pos_device&id=1&devicename=10000004&devicetype=beacon&devicex=12&devicey=11&rssi1m=-59&group=ground
参数说明
cmd:命令类型，固定是 update_pos_device
id:注意这个是表 id 不是设备 id，设备 id 不能改
devicename:起个名字，不是汉字都可以
devicetype:如果是基站设备这里填 station 如果是 beacon 就填 beacon
devicex:设备的 x 相对坐标
devicey:设备的 y 相对坐标
rssi1m:1 米处的 rssi 值，一般金桔的 beacon 是-59，这个校准值可以再调节
group:分组的组名，不能是汉字

3.2.3.4.清空位标数据

清空位置数据，删除所有配置的信息
http://127.0.0.1:8888/api/location?cmd=clear_pos_device

注意：此接口清空所有配置，使用时请慎用。

3.2.3.5. 删除位置数据

删除位置数据
http://127.0.0.1:8888/api/location?cmd=delete_pos_device&id=1
参数说明
cmd:命令类型，固定是 delete_pos_device
id:注意这个是表 id 不是设备 id

3.2.4. 位置服务 API 接口

1. 添加定位固定设备：
   接口：
   http://127.0.0.1:8888/api/location?cmd=add_pos_device?deviceid={deviceID}&devicename={deviceName}&devicetype={deviceType}&devicex={x}&devicey={y}&rssi1m={rssi}&group={group}

   参数说明：
   deviceid：需要添加的设备 ID
   deivicdname：需要添加的设备名称，主要用于方便查看和管理，可以设置成跟 id 一样
   devicetype：beacon 或者 station，设备类型
   devicex：设备的 x 轴坐标
   devicey：设备的 y 轴坐标
   rssi1m：添加默认的较准 rssi 值，如果不清楚这里填 -59，默认值
   group：分组名。
   详细含义参看上面的表格说明。
   例如，添加 major 为 88AA，minor 为 77EE 的 beacon 到(200,-200)位置系统中，
   http://127.0.0.1:8888/api/location?cmd=add_pos_device?deviceid=88AA77EE&devicename=88AA77EE&devicetype=beacon&devicex=200&devicey=-200&rssi1m=-59&group=floor1

   返回：{result：ok}，ok 即执行成功，fail 即执行失败
2. 更新定位固定设备：
   接口：
   http://127.0.0.1:8888/api/location?cmd=update_pos_device?id={ID}&devicename={deviceName}&devicetype={deviceType}&devicex={x}&devicey={y}&rssi1m={rssi}&group={group}

   参数说明：
   id：注意这个 ID 是数据库表 id，后面获取所有的设备时，有这个字段的值
   deivicdname：需要修改的设备名称，主要用于方便查看和管理，可以设置成跟 id 一样
   devicetype：beacon 或者 station，设备类型

devicex：设备的 x 轴坐标
devicey：设备的 y 轴坐标
rssi1m：修改默认的较准 rssi 值，如果不清楚这里填 -59，默认值
group：分组名。
详细含义参看上面的表格说明。
例如，修改表 id 为 2 的 beacon 到(200,-200)位置系统中，
http://127.0.0.1:8888/api/location?cmd=update_pos_device?id=2&devicename=88AA77EE&devicetype=beacon&devicex=200&devicey=-200&rssi1m=-59&group=floor1

返回：{result：ok}，ok 即执行成功，fail 即执行失败

3. 通过设备 id 删除设备
接口：
http://127.0.0.1:8888/api/location?cmd=delete_pos_device_bydeviceid?id={deviceID}
参数说明:
deviceid: 需要添加的设备 ID
例如，删除 major 为 88AA，minor 为 77EE 的 beacon，
http://127.0.0.1:8888/api/location?cmd=delete_pos_device_bydeviceid?id=88AA77EE
返回：{result：ok}，ok 即执行成功，fail 即执行失败
4. 通过表 id 删除设备
接口：
http://127.0.0.1:8888/api/location?cmd=delete_pos_device?id={ID}
参数说明:
id: 注意这个 ID 是数据库表 id，后面获取所有的设备时，有这个字段的值
例如，删除表 id 为 2 的设备，
http://127.0.0.1:8888/api/location?cmd=delete_pos_device?id=2
返回：{result：ok}，ok 即执行成功，fail 即执行失败

5. 清空设备
接口：
http://127.0.0.1:8888/api/location?cmd= clear_pos_device
返回：{result：ok}，ok 即执行成功，fail 即执行失败

6. 获取所有设备
接口：
http://127.0.0.1:8888/api/location?cmd=get_pos_devices
返回：
设备的列表数组，字段说明

Key	字段	含义
Id	表 ID	数据库中自增长的表 ID
DeviceId	设备 ID	如果添加的是 beacon 设备那么这个 ID 就是 MajorHex+MinorHex, 例如：一个 beacon 的 major 是 25517 那么 16 进制是 63AD（注意一定要大写），minor 是 63382，16 进制是 F796，那么这个设备的 id 就是：63ADF796。

		如果添加的基站设备，那么这个 ID 就是基站的 ID，例如 10000018
DeviceName	设备名称	方便管理，可以任意起名称（最好是字母数字）。
DeviceType	类型	是 BEACON 设备还是基站设备（值为：beacon 或 station）
DeviceX	X 坐标	设置设备的 X 坐标值（float 类型）
DeviceY	Y 坐标	设置设备的 Y 坐标值（float 类型）
Device1mRssi	1 米 RSSI	这个主要是计算位置时校准需要用到的参数，如果不了解可以使用默认值，当测量位置有误差时可以调整这个参考值例如：-59
DeviceGroup	分组名称	这个是对设备的分组，主要作用例如，当胸牌处于 1 楼楼梯口那么在此位置有可能取到 2 楼 beacon 的信号值，对于不在 2 楼的移动的胸牌来说这个设备有可能是增加误差的，所以对设备进行分组管理，那么在 1 楼的胸牌将过滤掉 2 楼的 beacon，有助于精准计算位置。（不能出现汉字）

例如：


```
[
    {
        "Id": 12,
        "DeviceId": "B74472FB",
        "DeviceName": "5",
        "DeviceType": "beacon",
        "DeviceX": "100",
        "DeviceY": "500",
        "Device1mRssi": "-59",
        "DeviceGroup": "floor1"
    },
    {
        "Id": 13,
        "DeviceId": "10000018",
        "DeviceName": "6",
        "DeviceType": "station",
        "DeviceX": "400",
        "DeviceY": "100",
        "Device1mRssi": "-59",
        "DeviceGroup": "floor1"
    },
    {
        "Id": 8,
        "DeviceId": "7F40506D",
        "DeviceName": "1",
```

```
        "DeviceType": "beacon",
        "DeviceX": "0",
        "DeviceY": "500",
        "Device1mRssi": "-59",
        "DeviceGroup": "floor1"
    },
    {
        "Id": 9,
        "DeviceId": "02A7C78B",
        "DeviceName": "2",
        "DeviceType": "beacon",
        "DeviceX": "500",
        "DeviceY": "500",
        "Device1mRssi": "-59",
        "DeviceGroup": "floor1"
    },
    {
        "Id": 10,
        "DeviceId": "7715C367",
        "DeviceName": "3",
        "DeviceType": "beacon",
        "DeviceX": "0",
        "DeviceY": "0",
        "Device1mRssi": "-59",
        "DeviceGroup": "floor1"
    },
    {
        "Id": 11,
        "DeviceId": "D22AE966",
        "DeviceName": "4",
        "DeviceType": "beacon",
        "DeviceX": "700",
        "DeviceY": "200",
        "Device1mRssi": "-59",
        "DeviceGroup": "floor1"
    }
]
```


3.3. ACserver 推送数据 API 接口

定位设备的网络管理软件（webmain）推送消息以 json 的对象形式对外提供数据，主要有 4 种类型，分别是：

location_stations，location_beacon，location_beacon_alarm，location_stations_alarm。

网络管理推送数据：通过我方网络管理软件（webmain）以 json 的对象形式提供的数据接口，包括以下四种：
Location_station：定位设备通过蓝牙传输的方式将定位位标数据推送至服务器的一种类型；
Location_beacon：定位设备通过 LoRa 传输的方式将定位位标数据推送至服务器的一种类型；
location_stations_alarm：定位设备通过蓝牙传输的方式将报警数据推送至服务器的一种类型；
location_beacon_alarm：定位设备通过 LoRa 传输的方式将报警数据推送至服务器的一种类型。

3.3.1. location_stations 类型数据说明

此类型数据是 webmain 将节点的蓝牙数据整理后推送给客户平台，当用户胸牌处于移动触发状态时发送。

字段	说明
FreOrChan	忽略，此类型数据无效。
Gateway	忽略，此类型数据无效。
Rssi	忽略，此类型数据无效。
Snr	忽略，此类型数据无效。
NodeId	胸牌的 Id
Type	消息类型，此类型是 location_stations
battery	电池电压值
Beacon_data	此数据类型里面是一个数组，是该定位胸牌搜索到当前周边的 beacon 数据，数组里每一个对象又是一个 json 对象对应每一个 beacon 的数据。 major：beacon 的 major 十进制的值 majorHex：beacon 的 major 十六进制的值 minor：beacon 的 minor 十进制的值 minorHex：beacon 的 minor 十六进制的值 rssi：胸牌搜索到 beacon 广播的信号强度
node_station_data	此数据类型里面是一个数组，是该定位胸牌搜索到当前周边的基站数据，数组里每一个对象又是一个 json 对象对应每一个基站的数据。 StationId：基站的 ID rssi:胸牌搜索到基站广播的信号强度
data	此数据类型里面是一个数组，此数据是胸牌周边基站收到胸牌的信号强度值，注意跟上面的字段区分，上面是胸牌收到基站的信号强度，此数据是基站搜索到胸牌的信号强度，数组里每一个对象又是一个 json 对象对应每一个基站的数据。 StationId：基站的 ID rssi:基站搜索到胸牌广播的信号强度

下面为该数据类型的实例：


```
{
    "FreOrChan": "0",
    "Gateway": "00000000",
    "NodeId": "997A4EDC",
    "Rssi": "0.0000",
    "Snr": "0.0000",
    "SystemId": "10990143",
    "Type": "location_stations",
    "battery": "3.817",
    "beacon_data": [
        {
            "major": "62232",
            "majorHex": "F318",
            "minor": "11537",
            "minorHex": "2D11",
            "rssi": "-77"
        },
        {
            "major": "14417",
            "majorHex": "3851",
            "minor": "34321",
            "minorHex": "8611",
            "rssi": "-65"
        },
        {
            "major": "3734",
            "majorHex": "0E96",
            "minor": "9763",
            "minorHex": "2623",
            "rssi": "-43"
        },
        {
            "major": "50117",
            "majorHex": "C3C5",
            "minor": "58839",
            "minorHex": "E5D7",
            "rssi": "-79"
        },
        {
            "major": "1",
            "majorHex": "0001",
            "minor": "12139",
```

```
            "minorHex": "2F6B",
            "rssi": "-54"
        },
        {
            "major": "1",
            "majorHex": "0001",
            "minor": "19282",
            "minorHex": "4B52",
            "rssi": "-57"
        },
        {
            "major": "1",
            "majorHex": "0001",
            "minor": "4572",
            "minorHex": "11DC",
            "rssi": "-56"
        },
        {
            "major": "34935",
            "majorHex": "8877",
            "minor": "2576",
            "minorHex": "0A10",
            "rssi": "-60"
        }
    ],
    "data": [
        {
            "rssi": -69,
            "station": "10000015"
        }
    ],
    "node_station_data": [
        {
            "rssi": "-72",
            "stationId": "10000015"
        }
    ]
}
```

3.3.2. location_stations_alarm 类型数据说明

此类型数据是 webmain 将节点的蓝牙报警数据整理后推送给客户平台，当用户在运行状态下按下按钮时发送。

字段	说明
FreOrChan	忽略，此类型数据无效。
Gateway	忽略，此类型数据无效。
Rssi	忽略，此类型数据无效。
Snr	忽略，此类型数据无效。
NodeId	胸牌的 Id
Type	消息类型，此类型是 location_stations_alarm
battery	电池电压值
press	报警按了几下按钮
Beacon_data	此数据类型里面是一个数组，是该定位胸牌搜索到当前周边的 beacon 数据，数组里每一个对象又是一个 json 对象对应每一个 beacon 的数据。 major：beacon 的 major 十进制的值 majorHex：beacon 的 major 十六进制的值 minor：beacon 的 minor 十进制的值 minorHex：beacon 的 minor 十进制的值 rssi：胸牌搜索到 beacon 广播的信号强度
node_station_data	此数据类型里面是一个数组，是该定位胸牌搜索到当前周边的基站数据，数组里每一个对象又是一个 json 对象对应每一个基站的数据。 StationId：基站的 ID rssi:胸牌搜索到基站广播的信号强度
data	此数据类型里面是一个数组，此数据是胸牌周边基站收到胸牌的信号强度值，注意跟上面的字段区分，上面是胸牌收到基站的信号强度，此数据是基站搜索到胸牌的信号强度，数组里每一个对象又是一个 json 对象对应每一个基站的数据。 StationId：基站的 ID rssi:基站收到胸牌广播的信号强度

下面为该数据类型的实例：


```
{
    "FreOrChan": "0",
    "Gateway": "00000000",
    "NodeId": "997A4EDC",
    "Rssi": "0.0000",
    "Snr": "0.0000",
    "SystemId": "10990143",
    "Type": "location_stations_alarm",
    "battery": "3.811",
```

```
"beacon_data": [
    {
        "major": "52028",
        "majorHex": "CB3C",
        "minor": "38412",
        "minorHex": "960C",
        "rssi": "-67"
    },
    {
        "major": "19958",
        "majorHex": "4DF6",
        "minor": "47957",
        "minorHex": "BB55",
        "rssi": "-39"
    },
    {
        "major": "16378",
        "majorHex": "3FFA",
        "minor": "5227",
        "minorHex": "146B",
        "rssi": "-69"
    },
    {
        "major": "58134",
        "majorHex": "E316",
        "minor": "5495",
        "minorHex": "1577",
        "rssi": "-71"
    },
    {
        "major": "34935",
        "majorHex": "8877",
        "minor": "2576",
        "minorHex": "0A10",
        "rssi": "-39"
    },
    {
        "major": "1",
        "majorHex": "0001",
        "minor": "19282",
        "minorHex": "4B52",
        "rssi": "-49"
    },
    {
```

```
            "major": "1",
            "majorHex": "0001",
            "minor": "4572",
            "minorHex": "11DC",
            "rssi": "-48"
        },
        {
            "major": "1",
            "majorHex": "0001",
            "minor": "12139",
            "minorHex": "2F6B",
            "rssi": "-42"
        }
    ],
    "data": [
        {
            "rssi": -50,
            "station": "10000015"
        }
    ],
    "node_station_data": [
        {
            "rssi": "-56",
            "stationId": "10000015"
        }
    ],
    "press": "1"
}
```


3.3.3. location_beacon 类型数据说明

此类型数据是 webmain 将节点的 LORA 数据整理后推送给客户平台，当用户胸牌处于移动触发状态时发送,注意该数据类型是当周边没有蓝牙基站的时候会发送，不是每次启动都会发送。

字段	说明
FreOrChan	基站接收到 Lora 信号的频率。
Gateway	接收到胸牌 lora 信号的 lora 基站的 ID。
Rssi	基站接收到 Lora 信号的强度。
Snr	基站接收到 Lora 信号的信噪比。
NodeId	胸牌的 Id。

Type	消息类型，此类型是 location_beacon。
battery	电池电压值
Beacon_data	此数据类型里面是一个数组，是该定位胸牌搜索到当前周边的 beacon 数据，数组里每一个对象又是一个 json 对象对应每一个 beacon 的数据。 major：beacon 的 major 十进制的值 majorHex：beacon 的 major 十六进制的值 minor：beacon 的 minor 十进制的值 minorHex：beacon 的 minor 十进制的值 rssi：胸牌搜索到 beacon 广播的信号强度
node_station_data	此数据类型里面是一个数组，是该定位胸牌搜索到当前周边的基站数据，数组里每一个对象又是一个 json 对象对应每一个基站的数据。 StationId：基站的 ID rssi:胸牌搜索到基站广播的信号强度

下面为该数据类型的实例：


```
{
    "FreOrChan": "481000000",
    "Gateway": "10000015",
    "NodeId": "997A4EDC",
    "Rssi": "121.0000",
    "Snr": "10.5000",
    "SystemId": "10990143",
    "Type": "location_beacon",
    "battery": "3.754",
    "beacon_data": [
        {
            "major": "58134",
            "majorHex": "E316",
            "minor": "5495",
            "minorHex": "1577",
            "rssi": "-70"
        },
        {
            "major": "52028",
            "majorHex": "CB3C",
            "minor": "38412",
            "minorHex": "960C",
            "rssi": "-62"
        },
        {
            "major": "18605",
            "majorHex": "48AD",
```

```
            "minor": "27148",
            "minorHex": "6A0C",
            "rssi": "-42"
        },
        {
            "major": "16378",
            "majorHex": "3FFA",
            "minor": "5227",
            "minorHex": "146B",
            "rssi": "-71"
        },
        {
            "major": "1",
            "majorHex": "0001",
            "minor": "4572",
            "minorHex": "11DC",
            "rssi": "-44"
        },
        {
            "major": "1",
            "majorHex": "0001",
            "minor": "19282",
            "minorHex": "4B52",
            "rssi": "-55"
        },
        {
            "major": "34935",
            "majorHex": "8877",
            "minor": "2576",
            "minorHex": "0A10",
            "rssi": "-37"
        },
        {
            "major": "1",
            "majorHex": "0001",
            "minor": "12139",
            "minorHex": "2F6B",
            "rssi": "-45"
        }
    ],
    "node_station_data": [
        {
            "rssi": "-58",
            "stationId": "10000015"
```

```
        }
    ]
}
```

3.3.4. location_beacon_alarm 类型数据说明

此类型数据是 webmain 将节点的 LORA 报警数据整理后推送给客户平台，当用户在运行状态下按下按钮时发送。

字段	说明
FreOrChan	基站接收到 Lora 信号的频率。
Gateway	接收到胸牌 lora 信号的 lora 基站的 ID。
Rssi	基站接收到 Lora 信号的强度。
Snr	基站接收到 Lora 信号的信噪比。
NodeId	胸牌的 Id。
Type	消息类型，此类型是 location_beacon_alarm。
battery	电池电压值
press	报警按了几下按钮
Beacon_data	此数据类型里面是一个数组，是该定位胸牌搜索到当前周边的 beacon 数据，数组里每一个对象又是一个 json 对象对应每一个 beacon 的数据。 major：beacon 的 major 十进制的值 majorHex：beacon 的 major 十六进制的值 minor：beacon 的 minor 十进制的值 minorHex：beacon 的 minor 十进制的值 rssi：胸牌搜索到 beacon 广播的信号强度
node_station_data	此数据类型里面是一个数组，是该定位胸牌搜索到当前周边的基站数据，数组里每一个对象又是一个 json 对象对应每一个基站的数据。 StationId：基站的 ID rssi:胸牌搜索到基站广播的信号强度

下面为该数据类型的实例：


```
{
    "FreOrChan": "482000000",
    "Gateway": "10000015",
    "NodeId": "997A4EDC",
    "Rssi": "89.0000",
    "Snr": "10.7500",
    "SystemId": "10990143",
    "Type": "location_beacon_alarm",
    "battery": "3.800",
    "beacon_data": [
```

```
{
    "major": "16378",
    "majorHex": "3FFA",
    "minor": "5227",
    "minorHex": "146B",
    "rssi": "-80"
},
{
    "major": "19958",
    "majorHex": "4DF6",
    "minor": "47957",
    "minorHex": "BB55",
    "rssi": "-58"
},
{
    "major": "52028",
    "majorHex": "CB3C",
    "minor": "38412",
    "minorHex": "960C",
    "rssi": "-75"
},
{
    "major": "58134",
    "majorHex": "E316",
    "minor": "5495",
    "minorHex": "1577",
    "rssi": "-86"
},
{
    "major": "34935",
    "majorHex": "8877",
    "minor": "2576",
    "minorHex": "0A10",
    "rssi": "-67"
},
{
    "major": "1",
    "majorHex": "0001",
    "minor": "19282",
    "minorHex": "4B52",
    "rssi": "-61"
},
{
    "major": "1",
```

```
            "majorHex": "0001",
            "minor": "12139",
            "minorHex": "2F6B",
            "rssi": "-68"
        },
        {
            "major": "1",
            "majorHex": "0001",
            "minor": "4572",
            "minorHex": "11DC",
            "rssi": "-64"
        }
    ],
    "node_station_data": [
        {
            "rssi": "-87",
            "stationId": "10000015"
        }
    ],
    "press": "1"
}
```


3.3.5. location_pos 类型数据说明

此类型数据是 webmain 将节点的 beacon 和基站信号数据计算成位置数据后推送给客户平台。

字段	说明
FreOrChan	忽略，此类型数据无效。
Gateway	忽略，此类型数据无效。
Rssi	忽略，此类型数据无效。
Snr	忽略，此类型数据无效。
NodeId	胸牌的 Id
Type	消息类型，此类型是 location_pos
Group	基于当前哪个 group 计算的位置，具体含义看配置说明。
GroupNumber	当前 Group 下有几组有效数据

Beacon_data	此数据类型里面是一个数组，是该定位胸牌搜索到当前周边的 beacon 数据，数组里每一个对象又是一个 json 对象对应每一个 beacon 的数据。 major：beacon 的 major 十进制的值 majorHex：beacon 的 major 十六进制的值 minor：beacon 的 minor 十进制的值 minorHex：beacon 的 minor 十进制的值 rssi：胸牌搜索到 beacon 广播的信号强度
position	经过计算后的位置坐标，里面含有 X，Y 的值

下面为该数据类型的实例：


```
{
    "FreOrChan": "0",
    "Gateway": "00000000",
    "NodeId": "10000004",
    "Rssi": "0.0000",
    "Snr": "0.0000",
    "SystemId": "10990143",
    "Type": "location_pos",
    "group": "floor1",
    "groupNumber": 4,
    "position": {
        "x": 442.77470115871535,
        "y": 294.3878062308057
    }
}
```


3.3.6. beacon_device 类型数据说明

此类型数据是将 beacon 类型的设备数据后推送给客户平台。

字段	说明
FreOrChan	忽略，此类型数据无效。
Gateway	忽略，此类型数据无效。
Rssi	忽略，此类型数据无效。
Snr	忽略，此类型数据无效。

NodeId	beacon 的 Id
Type	消息类型，此类型是 beacon_device
Beacon_service	Beacon 服务类型，比如 0215 是 ibeacon
Vendor_id	Beacon 的服务商 id，由蓝牙联盟提供，比如苹果为 004C
Beacon_uuid	Beacon 的 uuid 值
Rssi_level	Bedacon 的信号强度较准值，这个由 beacon 厂商设置，一般可以改。
Major	beacon 的 major 十进制的值
majorHex	beacon 的 major 十六进制的值
Minor	eacon 的 minor 十进制的值
minorHex	beacon 的 minor 十进制的值
data	此数据类型里面是一个数组，此数据是 beacon 周边基站收到 beacon 的信号强度值，数组里每一个对象又是一个 json 对象对应每一个基站的数据。 StationId：基站的 ID rssi:基站收到 beacon 广播的信号强度

下面为该数据类型的实例：


```
{
    "FreOrChan": "0",
    "Gateway": "00000000",
    "NodeId": "10000005",
    "Rssi": "0.0000",
    "Snr": "0.0000",
    "SystemId": "10990143",
    "Type": "beacon_device",
    "beacon_service": "0215",
    "beacon_uuid": "01122334455667788 99AABBC10000005",
    "data": [
        {
            "rssi": -62,
            "station": "10000018"
        },
        {
            "rssi": -58,
            "station": "10000017"
        }
    ],
    "major": "34935",
    "majorHex": "8877",
    "minor": "1000",
```

```
    "minorHex": "03E8",
    "rssi_level": "-50",
    "vendor_id": "0059"
}
```


3.3.7. location_stations_lowpower 类型数据说明

此类型数据是电压低于 3.4v 时，胸牌自动切换成低功耗模式，此模式下，胸牌可以持续运行 6 个月以上，系统收到此类型数据时，需要提醒用户给胸牌充电。

字段	说明
FreOrChan	忽略，此类型数据无效。
Gateway	忽略，此类型数据无效。
Rssi	忽略，此类型数据无效。
Snr	忽略，此类型数据无效。
NodeId	胸牌的 Id
Type	消息类型，此类型是 location_stations_lowpower
battery	当前电池电压值
data	此数据类型里面是一个数组，此数据是胸牌周边基站收到胸牌的信号强度值，数组里每一个对象又是一个 json 对象对应每一个基站的数据。 StationId：基站的 ID rssi:基站收到胸牌广播的信号强度

下面为该数据类型的实例：


```
{
    "FreOrChan": "0",
    "Gateway": "00000000",
    "NodeId": "10000004",
    "Rssi": "0.0000",
    "Snr": "0.0000",
    "SystemId": "10990143",
    "Type": "location_stations_lowpower",
    "battery": "3.380",
    "data": [
        {
            "rssi": -92,
            "station": "10000018"
        }
    ]
```

```
}
```

3.3.8. station_heart 类型数据说明

此类型数据是基站的心跳数据，当 webmain 下发轮询信息时，基站上传心跳信息。

字段	说明
FreOrChan	忽略，此类型数据无效。
Gateway	基站的 ID。
Rssi	忽略，此类型数据无效。
Snr	忽略，此类型数据无效。
NodeId	忽略，此类型数据无效
Type	消息类型，此类型是 station_heart
ip	当前心跳基站的 ip
port	当前心跳基站的端口号
version	当前心跳基站的版本号

下面为该数据类型的实例：


```
{
    "FreOrChan": "0",
    "Gateway": "10000018",
    "NodeId": "00000000",
    "Rssi": "0.0000",
    "Snr": "0.0000",
    "SystemId": "10990143",
    "Type": "station_heart",
    "ip": "192.168.1.140",
    "port": "10604",
    "version": "V-03-01-07"
}
```


3.3.9. station_start 类型数据说明

此类型数据是基站的启动事件数据，当基站启动或者重新启动时，基站上传此信息。

字段	说明
FreOrChan	忽略，此类型数据无效。
Gateway	基站的 ID。

Rssi	忽略，此类型数据无效。
Snr	忽略，此类型数据无效。
NodeId	忽略，此类型数据无效
Type	消息类型，此类型是 station_start
ip	当前启动基站的 ip
port	当前启动基站的端口号
version	当前启动基站的版本号

下面为该数据类型的实例：


```
{
    "FreOrChan": "0",
    "Gateway": "10000018",
    "NodeId": "00000000",
    "Rssi": "0.0000",
    "Snr": "0.0000",
    "SystemId": "10990143",
    "Type": "station_start",
    "ip": "192.168.1.140",
    "port": "10604",
    "version": "V-03-01-07"
}
```


3.3.10. 报警信息取消接口

调用：
http://127.0.0.1:8888/api/location?cmd=clear_alarm_all

备注：报警信息取消，为应用系统取消当前所有紧急报警。

3.4. 网络串口数据接口

网络串口数据接口是提供给高级开发者的开放式接口，采用此接口需自行解决定位业务过程中所涉及的所有网络及数据业务。我方仅提供数据通路管理，采用此接口要慎重评估自身开发能力及工作量。

此类接口使用时，请不用同时使用我方的 ACserver，若同时使用会有数据不能完全接收的风险。

使用此接口的特殊说明：

1、 我方不提供针对此接口的技术支持。

2、 接口过程中涉及的校验、数据位数请参照例程自行计算，若数据不准确，请仔细检查，我方不提供校验服务。
若使用过程中，对任何接口有疑问，请打开 ACserver 自行对照。

3.4.1. 基础说明

端口说明:
网关默认数据传输方式为：UDP
网关默认数据入站端口为：10352
网关默认数据出站端口为：10604
请在调试前，打开服务相关端口，若冲突请调整。

LoRa 默认频率:
上行：480.6~482MHz 8 个频段 ， 200KHz 一个步进。
下行： 480MHz

3.4.2. 网关事件类型数据格式

基站通过 udp 网络上传数据给服务器，报告基站的状态数据。
基础数据类型:

帧头	基站 ID	命 令 类型	长度	PAYLOAD	流水号	CRC 校验	帧尾
1 字节	4 字节	1 字节	1 字节	N 字节	2 字节	1 字节	1 字节

帧头：固定为 0x8E
基站 ID：贴在基站壳子上的 id，为 4 字节的 BCD 码，比如基站 id 为 10000017，这四个字节为 0x10,0x00,0x00,0x17
命令类型：表示这个串的含义，不同的命令类型，有不同的长度和解析，后面有详解。
长度：PAYLOAD 的长度。
PAYLOAD：负载的数据。
流水号：基站会自动累加流水号。
Crc 校验：从基站 ID 开始每个字节亦或到流水号，最后的到的字节放在这里(bcc 校验)。
帧尾：固定为 0x8D

3.4.2.1. 配置返回确认命令

确认命令类型：0x0A
该命令是对 udp 下发的基站配置命令的返回确认以表示配置成功，该命令的 Payload 里含有三个字节代表了基站的固件主版本号，这个 PAYLOAD 也可以忽略。
例如：0x8E10000017A003030103D20B7C8D 其中 030103 为版本号

3.4.2.2. 网络寻找基站命令确认

确认命令类型：0x96

该命令类型是网络上有平台发送 upd 广播来寻找基站回应时，基站上传的回应信息，该命令也经常用于基站心跳信息。该命令的 Payload 里含有三个字节代表了基站的固件主版本号，这个 PAYLOAD 也可以忽略。

例如：0x8E1000001796030301 03D70B4F8D 其中 030103 为版本号

3.4.2.3. 基站启动命令

命令类型：0X95

基站启动时，发送的命令。该命令的 Payload 里含有三个字节代表了基站的固件主版本号，这个 PAYLOAD 也可以忽略。

例如：0x8E100000179503030103010030100918D 其中 030103 为版本号

3.4.2.4. 读取基站配置返回数据

命令类型：0x99

该命令类型是网络上有平台询问该基站的配置信息，基站返回给平台的命令类型。

下表是该返回 payload 的详细解释。

字节号	解释
0~3，4 字节	基站的指向 ip，如果是广播就是四个 0，例如 0，0，0，0
4~7，4 字节	1301 的 0 通道的频率，例如 481000000，0x1C,0xAB,0x7A,0x40
8~11，4 字节	1301 的 1 通道的频率，例如 481000000，0x1C,0xAB,0x7A,0x40
12~21, 10 字节	1301 的通道掩码（详见 1301 手册），例如，1,1,1,1,1,1,1,1,1,1
22~31, 10 字节	1301 的射频通道链（详见 1301 手册），例如，1,1,1,0,0,0,0,0,0,1
32~35，4 字节	通道链 0 的频率区间（详见 1301 手册），例如-200
36~39，4 字节	通道链 1 的频率区间（详见 1301 手册），例如-200
40~43，4 字节	通道链 2 的频率区间（详见 1301 手册），例如-200
44~47，4 字节	通道链 3 的频率区间（详见 1301 手册），例如-200
48~51，4 字节	通道链 4 的频率区间（详见 1301 手册），例如-200
52~55，4 字节	通道链 5 的频率区间（详见 1301 手册），例如-200
56~59，4 字节	通道链 6 的频率区间（详见 1301 手册），例如-200
60~63，4 字节	通道链 7 的频率区间（详见 1301 手册），例如-200
64~67，4 字节	通道链 8 的频率区间（详见 1301 手册），例如-200
68~71，4 字节	通道链 9 的频率区间（详见 1301 手册），例如-200
72~79，8 字节	通道的扩频因子掩码（详见 1301 手册），例如 126,126,126,126,126,126,126,126
80, 1 字节(len)	指向 url 长度，如果为 0，就是按之前的指向 ip
81~（81+len）,len	url

字节	
（82+len）~（87+len），6字节	基站的 mac 地址，例如 0.8.220.18.171.205
（88+len）~（91+len），4字节	基站的 ip 地址，例如 192.168.1.105
（92+len）~（95+len），4字节	基站的地址掩码，例如 255.255.255.0
（96+len）~（99+len），4字节	基站的网关地址，例如 192.168.1.1
（100+len），1字节	是否是静态 ip，如果 1 就是静态 ip，0 是动态 ip

例如读取 IP 配置，

0x8E1000001799640000000001CAB7A401CBABC8001010101010101010101010100000000000000001FFF9E580FFFCF2C000000000FFF9E580FFFCF2C00000000000030D4000061A8000000000000000007E7E7E7E7E7E7E7E0010000017ABCDC0A80269FFFFFF00C0A8010101D10B118D

3.4.3. 蓝牙服务消息

基站的蓝牙服务的相关消息，这里面主要分成三种数据，按 payload 的第一个字节（payload[0]）来区分。

a. 蓝牙连接消息：payload 为：0x01(连接消息类型),0x00，0x20（2 个字节 gatt 消息长度），0x00，0x00，0x00，0x00，0x00，0x01(6 个字节 mac 地址)
b. 蓝牙断开连接消息：payload 为：0x02(断开连接消息类型), 0x00，0x00，0x00，0x00，0x00，0x01(6 个字节 mac 地址)，0x22(1 个字节的断开原因)
c. 蓝牙超时消息：payload 为：0x04(超时消息类型), 0x00，0x00，0x00，0x00，0x00，0x01(6 个字节 mac 地址)，0x00(1 个字节代表是否有连接)
d. 蓝牙接收消息：payload 为：0x03(连接消息类型), 0x00，0x00，0x00，0x00，0x00，0x01(6 个字节 mac 地址)，0x00,0x00,0x00(如果收到 3 个数就是蓝牙收到 3 个数)

3.4.4. 收到节点蓝牙广播消息

命令号：0xB0

该消息内容里可能含有多个节点的广播数据，所以 payload 第一个字节 payload[0]代表这个数据包里含有多少个节点的数据，那么从第二个字节 payload[1]开始按下面的协议规则排列（后面循环按这个结构）。

在 payload 里字节位置	解释（若长度超过 255，实际长度-11）
1~4，4 字节	节点的 id，例如 0x10,0x00,0x00,0x12,节点 id 是 10000012
5，1 字节	节点广播的 rssi，例如-80
6，1 字节	节点广播内容的命令类型，例如，定位是 0x20
7，1 字节	广播内容长度
8~n，n-7 个字节	广播内容，长度按上面的字节数，payload 部分

例如，蓝牙报警类型
基站上传数据:
0x8E10000017B03C0110000004BF2034040088C2407DAC011B4A6099DC0136639163B601C48C2173BD01B3A72FBCB6012A76CBD4B901F3725E9DA90110000017C602CD0E8A1E9D8D
下面是对 payload 部分字节的解析，头尾部分请看上表:

0x8E10000017B03C
（下面开始是 payload，第一位是 01，代表只有一个节点的广播数据）
01(代表有一个数据)
10000004(1~4 字节代表 ID)
BF(rssi 的 INT8 类型)
20（是胸牌的广播类型）
34（后面数据长度）
04（报警蓝牙广播类型，05 是低电量）
00（保留位）
（后面每六个字节代表一个 beacon，如果 beacon 类型是 01 就是 beacon,如果 beacon 类型是 02 就是基站）
88C2（beacon 的 Major）
407D（beacon 的 Minor）
AC（RSSI 值 int8 类型）
01（beacon 类型，01 是 beacon）
(后面每六个都是按上面得解析)
1B4A6099DC01（6 字节 beacon 数据）
36639163B601（6 字节 beacon 数据）
C48C2173BD01（6 字节 beacon 数据）
B3A72FBCB601（6 字节 beacon 数据）
2A76CBD4B901（6 字节 beacon 数据）
F3725E9DA901（6 字节 beacon 数据）
10000017（这个是基站 ID）
C6（RSSI 值 int8 类型）
02（beacon 类型，02 就是基站）
CD0E（两个字节是电池电量）
（payload 截至）
8A1E9D8D

下面是蓝牙普通位置信息上传:

0x8E10000017B04D0110000004C62045033F20621ADB012730FAE8D701271B271BD90127C
D7AB4D30127426B1AD7013F20C169D1010001F037EA01D29149F6CF012730FAECE001000
10107D00110000017CE02CD0E1F7AEE8D
下面是对 payload 部分字节的解析，头尾部分请看上表：
0x8E10000017B04D
（下面开始是 payload，第一位是 01，代表只有一个节点的广播数据）
01(代表有一个数据)
10000004(1~4 字节代表 ID)
C6(rssi 的 INT8 类型)
20（是胸牌的广播类型）
45（后面数据长度）
03（蓝牙普通位置信息广播类型）（
后面每六个字节代表一个 beacon，如果 beacon 类型是 01 就是 beacon,如果 beacon 类型是 02 就是基站）
3F20（beacon 的 Major）
621A（beacon 的 Minor）
DB（RSSI 值 int8 类型）
01（beacon 类型，01 是 beacon）
（后面每六个都是按上面得解析）
2730FAE8D701（6 字节 beacon 数据）
271B271BD901（6 字节 beacon 数据）
27CD7AB4D301（6 字节 beacon 数据）
27426B1AD701（6 字节 beacon 数据）
3F20C169D101（6 字节 beacon 数据）
0001F037EA01（6 字节 beacon 数据）
D29149F6CF01（6 字节 beacon 数据）
2730FAECE001（6 字节 beacon 数据）
00010107D001（6 字节 beacon 数据）
10000017（这个是基站 ID）
CE（RSSI 值 int8 类型）
02（beacon 类型，02 就是基站）
CD0E（两个字节是电池电量）
（payload 截至）
1F7AEE8D

3.4.5. LoRa 传输定位数据格式

命令号：0x80
基站的 lora 数据上传跟事件型稍有不同，此类型的命令类型为 0x80 固定，但是有相应的 lora 信息上传，所以要单独解释此类数据，接收到数据在 payload 里。

字节位	解释
0，1 字节	帧头，固定为 0x8E
1~4，4 字节	基站 id

5，1 字节	命令类型，在这里是固定 0x80
6~9，4 字节	接收到的 lora 数据的频率，semtech 官方说明：central frequency of the IF chain
10，1 字节	semtech 官方说明： by which IF chain was packet received
11，1 字节	semtech 官方说明： status of the received packet
12~15，4 字节	semtech 官方说明： internal concentrator counter for timestamping, 1 microsecond resolution
16，1 字节	semtech 官方说明： through which RF chain the packet was received
17，1 字节	semtech 官方说明： modulation used by the packet
18，1 字节	调制带宽，semtech 官方说明： modulation bandwidth (LoRa only)
19~22，4 字节	数据包的 Rx 数据率（Lora 为 SF）semtech 官方说明： RX datarate of the packet (SF for LoRa)
23，1 字节	数据包的纠错码，semtech 官方说明：error-correcting code of the packet (LoRa only)
24~27，4 字节	平均信号强度（单位：db），semtech 官方说明：average packet RSSI in dB
28~31，4 字节	平均包信噪比，semtech 官方说明：average packet SNR, in dB (LoRa only)
32~35，4 字节	最小包信噪比，semtech 官方说明：minimum packet SNR, in dB (LoRa only)
36~39，4 字节	最大包信噪比，semtech 官方说明：maximum packet SNR, in dB (LoRa only)
40~41，2 字节	包校验，semtech 官方说明：CRC that was received in the payload
42~43，2 字节	payload 长度，高字节在前
44~n，n-41 字节	Payload，lora 接收到的数据
N+1~n+3, 2 字节	流水号
N+4，1 字节	校验位，从基站 id 到流水号的亦或值（bcc 校验）
N+5，1 字节	0x8D 帧尾

例如

0x8E10000017801CA86D0004104ABCE09B00100300000000402430B0000413C0000410C0000418C0000C6D10040

AF10000004A0350604002A76CBD4B80188C2407DAD01C48C2173BD011B4A6099DC01B3A72FBCB901F3725E9DA90136639163B80110000017C602CD0EE200583C（这一段是 payload）1A1DA48D

8E（帧头）

12 78 00 65（4 字节网关 ID）

80（命令）

1C BA BC 80 lora 官方说明：central frequency of the IF chain

02 lora 官方说明： by which IF chain was packet received

10 lora 官方说明： status of the received packet 接收数据包的状态

63 E0 6B BC lora 官方说明： internal concentrator counter for timestamping, 1 microsecond resolution

01 lora 官方说明： through which RF chain the packet was received
10 lora 官方说明： modulation used by the packet
03 lora 官方说明： modulation bandwidth (LoRa only) 调制带宽
00 00 00 10 lora 官方说明： RX datarate of the packet (SF for LoRa) 数据包的 Rx 数据率（Lora 为 SF）
02 lora 官方说明：error-correcting code of the packet (LoRa only) 数据包的纠错码
42 CA 00 00 lora 官方说明：average packet RSSI in dB 平均包 rssi（单位：db）
40 70 00 00 lora 官方说明：average packet SNR, in dB (LoRa only) 平均包信噪比
3F 80 00 00 lora 官方说明：minimum packet SNR, in dB (LoRa only) 最小包信噪比
41 5C 00 00 lora 官方说明：maximum packet SNR, in dB (LoRa only) 最大包信噪比
4E 6C lora 官方说明：CRC that was received in the payload
0F（这条数据的长度，范围：AF 开始至序列号之前）
AF/12 78 04 34（标签 ID）
数据
（AF：帧头

60 00 00 19：标签 ID

A0：定位数据
3D：长度（范围：类型开始至电池电量）

02 ： 数据类型

6B CB major
05 E4 minor
CE rssi
这样九个九个字节循环 一直到 最后剩 10 个字节

01(beacon 类型可忽略)
79 0E 电池电量

AD 09 两个字节流水号

80 bcc 校验（标签 ID 开始至流水号结束）

3C 帧尾）
00 0A（序列号）
76（校验/使用 BBC 校验算法/校验范围：标签 ID 开始序列号结束）
8D（帧尾）

3.4.6. LoRa 传输心跳包数据格式

8E 13 01 10 32 97 03 05 00 01 00 02 A2 8D
心跳包格式:
8E（一字节，帧头）
13 01 10 32（四字节，基站 ID）
97 （一字节，命令类型）
01（一字节通道号，可忽略）
00（一字节 SNR，可忽略）
00 00（两字节 RSSI，可忽略）
03 （一字节 payload 长度）
05 00 01 （三字节固件版本号）
00 11 （两字节流水号）
B1 （一字节 BCC 校验）
8D（一字节帧尾）

3.4.7. LoRa 传输报警数据格式

8E（帧头）
12 78 00 65（4 字节网关 ID）
80（命令）
1C BA BC 80 lora 官方说明：central frequency of the IF chain
02 lora 官方说明： by which IF chain was packet received
10 lora 官方说明： status of the received packet 接收数据包的状态
63 E0 6B BC lora 官方说明： internal concentrator counter for timestamping, 1 microsecond resolution
01 lora 官方说明： through which RF chain the packet was received
10 lora 官方说明： modulation used by the packet
03 lora 官方说明： modulation bandwidth (LoRa only) 调制带宽
00 00 00 10 lora 官方说明： RX datarate of the packet (SF for LoRa) 数据包的 Rx 数据率（Lora 为 SF）
02 lora 官方说明：error-correcting code of the packet (LoRa only) 数据包的纠错码
42 CA 00 00 lora 官方说明：average packet RSSI in dB 平均包 rssi（单位：db）
40 70 00 00 lora 官方说明：average packet SNR, in dB (LoRa only) 平均包信噪比
3F 80 00 00 lora 官方说明：minimum packet SNR, in dB (LoRa only) 最小包信噪比
41 5C 00 00 lora 官方说明：maximum packet SNR, in dB (LoRa only) 最大包信噪比
4E 6C lora 官方说明：CRC that was received in the payload
0F（这条数据的长度，范围：AF 开始至序列号之前）
AF/12 78 04 34（标签 ID）
数据
（AF 帧头

60 00 00 15 胸牌 id

A0 定位数据

11 长度（范围：类型开始至电池电量）
04 定位数据的协议类型 04 是报警（05 是低电量）
00 01 报警前最近一次搜到的 major 值
FE 05 报警前最近一次搜到的 minor 值
CB 报警前最近一次搜到的 rssi
01 按了多少下
8E 0E 电池电量
11 05 序列流水号
62 bcc 校验（校验范围从标签 ID 开始至序列号结束）
3C 帧尾）
00 0A（序列号）
76（校验/使用 BBC 校验算法/校验范围：标签 ID 开始序列号结束）
8D（帧尾）

3.4.8. 静止标志位

静止后至最终不再发送数据时会发送一次静止标志位，判断标签已经进入静止状态，标志位为下面数据表中的 0A。
8E 13 01 10 32 80 1C BA BC 80 02 10 0C C1 AF 84 01 10 03 00 00 00 20 02 43 07 00 00 41 10 00 00 40 D8 00 00 41 48 00 00 AF A6 00 14 AF 10 00 00 05 A0 09 0A 00 01 F4 6E C7 00 92 0F 90 00 E7 3C 00 23 C7 8D

3.4.9. Payload 数据

AF：帧头

60 00 00 19：标签 ID

A0：定位数据
3D：长度（范围：类型开始至电池电量）

02 ： 数据类型

6B CB major
05 E4 minor
CE rssi
这样九个九个字节循环 一直到 最后剩 10 个字节

01(beacon 类型可忽略)

79 0E 电池电量

AD 09 两个字节流水号

80 bcc 校验（标签 ID 开始至流水号结束）

3C 帧尾）

第4章 iBeacon 定位信标接口

金桔 beacon 有两种运行模式，定位锚点运行模式和标准 iBeacon 运行模式。
定位锚点运行模式下，金桔 beacon 将交替广播标准 iBeacon 数据和定位锚点需要上报的内容数据，上报的内容数据中包含电量信息等。

注意：标准 iBeacon 中采用的是苹果公司的标准协议，在此协议中，无任何数据可更改，若单独使用 iBeacon 模式不能进行配置、读取电量等操作。
金桔 beacon 以 IB 开头，此标志位若更改，则不能采用金桔官方工具：金桔 BOX 进行配置，需自行部署。

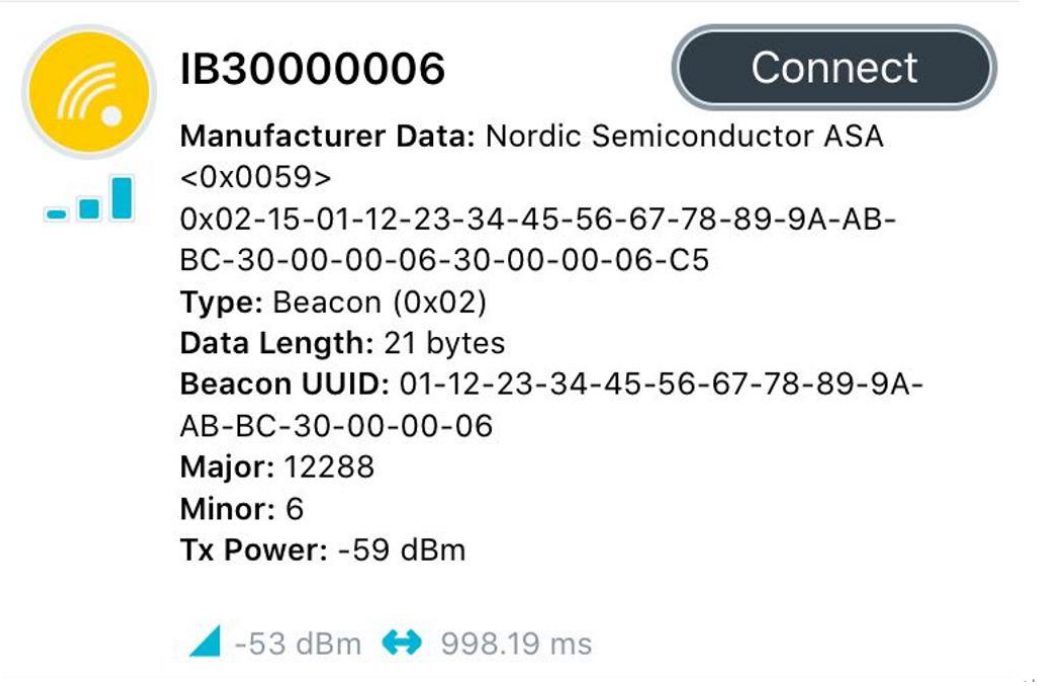
4.1. 金桔 beacon 数据协议

以上为通过 nrf connect 读取到的金桔 beacon 的广播数据，一共 11 个字节。
字节 0：厂家代码，这里是 0x06 金桔 beacon 固定的代码
字节 1：广播长度，0x15，这个是广播总长度
字节 2~3：是电量信息，高位在前，这里是 0c c5 即电量为 0x0cc5，十进制是 3269，即 3.269V 电压
字节 4~5：是 beacon 的 major，这里是 30 00 即 0x3000 十进制就是 12288（金桔 beacon 的出厂默认设置，major 是 id 的高两位，minor 是 id 的低两位）
字节 6~7：是 beacon 的 minor，这里是 00 02 即 0x0002 十进制就是 2（金桔 beacon 的出

厂默认设置，major 是 id 的高两位，minor 是 id 的低两位）
字节 8~9：是 beacon 的广播周期，高位在前，这里是 01 F4 即 0x01F4，十进制就是 500 广播周期就是 500ms
字节 10：是 beacon 的发送功率，这里是 0，即 0x00 beacon 的蓝牙发送功率是 0dbm

4.2. 标准 iBeacon 数据协议

标准 iBeacon 运行模式将按 iBeacon 标准数据格式，无电量等信息。如下图：



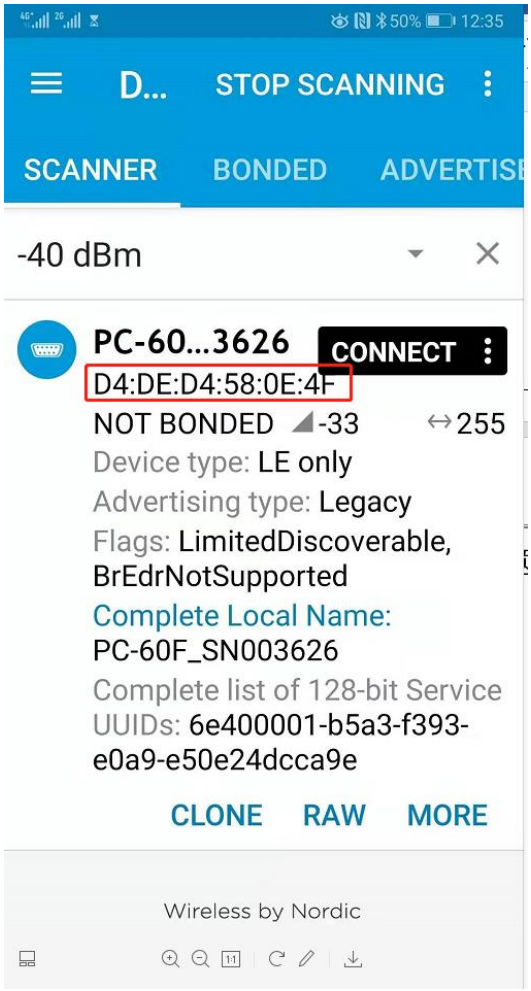
第5章 蓝牙网关透传数据接口

金桔蓝牙网关均可支持与市场上蓝牙 4.0 及以上协议（蓝牙 4.1、蓝牙 4.2、蓝牙 5.0、蓝牙 longrange）的设备进行双向数据交互。此类设备只要设备本身符合以下标准，可自行开发实现数据交互：
1、目前支持标准 nus 服务。其他类型蓝牙协议需定制开发；
2、 MAC 地址需要公开，不能隐藏。
若对方设备符合以上两个条件，请用小程序将网关配置为透传模式即可进行实现数据交互。
此接口是基于 ACserver 完成，若采用透传模式，请自行定义协议，我方不提供技术支持。

5.1. 蓝牙网关数据透传基本说明

5.1.1. 设备 MAC 地址查看

苹果手机请用手机到应用商店下载 NRF connect 工具；
安卓手机请联系厂商索要 APK。



注意：上图中手机读取的 MAC 地址与网关下发的设备 MAC 地址是反向的。
手机读取为：D4:DE:D4:58:0E:4F
数据下发应该为：4F, 0E, 58, D4, DE, D4

5.1.2. 数据连接能否支持被动连接

目前的数据连接不支持被动连接，即：若服务器未发起主动连接，网关不会推送节点的 MAC 地址到服务器。

5.1.3. 数据连接是否可以支持底层协议

目前对数据对接未开放底层协议，请用 webmain 进行连接。

5.1.4. 设备向网关发送数据存在粘包

设备<-发生粘包->网关
网关在此处理是完全透传，请先用网络调试助手查看是否存在粘包，若是设备发给网关的数据粘包，我方无法处理，请自行合并解析。

5.2. 蓝牙网关数据透传连接指令

5.2.1. 设备连接指令

[{"command":"41","gatewayId":"10000015","target":"gateway","contentType":"byte",
"content":"4F,0E,58,D4,DE,D4,F4,01,00,00"}]
gatewayId:填入基站 id
command：固定
target：固定
contentType：固定
content：逗号分开的 16 进制，逗号是英文逗号,一共 10 个字节，前 6 位是 mac 地址，注意 mac 顺序，跟手机上正好反过来，后四个是超时时间，低字节在前，比如 0a，00，00，00 就是 10 秒。
注意：时间为持续连接时间，若在此时间内未有设备连接，系统自动退出。需再次申请连接。

5.2.2. 连接成功返回指令

{"FreOrChan":"0","Gateway":"10000015","NodeId":"4F0E58D4","Rssi":"0.0000","Snr":"0.0000","
SystemId":"10990143","Type":"ble_evt_rxd","data":"D0","mac":"4F0E58D4DED4"}
连接上，数据返回，收到的数据　是 ble_evt_rxd 类型

【2019-11-18 13:06:55:431】{"FreOrChan":"0","Gateway":"10000015","NodeId":"4F0E58D4","Rssi":"0.0000","Snr":"0.0000","SystemId":"10990143","Type":"ble_evt_connected","data_length":"64","mac":"4F0E58D4DED4"}

5.2.3. 连接失败返回指令

{"FreOrChan":"0","Gateway":"10000015","NodeId":"00000000","Rssi":"0.0000","Snr":"0.0000","
SystemId":"10990143","Type":"ble_evt_timeout","mac":"000000000000","status":"disconnecte
d"}
【2019-11-18 12:11:50:012】

连接失败，数据返回，收到的数据 是 ble_evt_timeout 类型,status 结果为：Disconnected

【2019-11-18 12:52:35:702】{"FreOrChan":"0","Gateway":"10000015","NodeId":"00000000","Rssi":"0.0000","Snr":"0.0000","SystemId":"10990143","Type":"ble_evt_timeout","mac":"000000000000","status":"disconnected"}

5.3. 数据连接示例

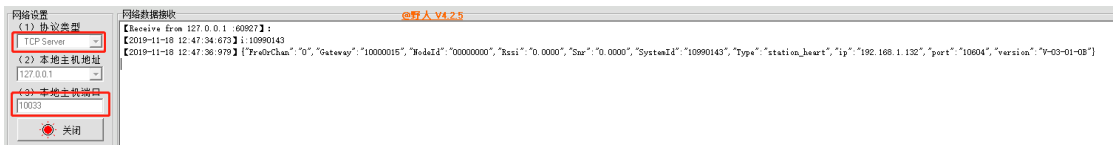
1、 在 ACserver 中配置网关，启动数据透传功能，参数如下：
注意：蓝牙接收功能、广播透传功能必须开启。

2、配置推送
在系统信息——推送接口

注意：设置推送时，消息类型过滤、消息字段过滤无需过滤（高级配置可选，测试无需选择）

3、打开网络调试助手

打开助手后可以收到网关的心跳



4、根据对接协议发送数据包

发送数据包

4、将设备调整为可连接状态

在设备连接期限内打开设备，若超出实现，系统会返回 disconnected，需再次发起建立连接。

【2019-11-18 12:52:35:702】{"FreOrChan":"0","Gateway":"10000015","NodeId":"00000000","Rssi":"0.0000","Snr":"0.0000","SystemId":"10990143","Type":"ble_evt_timeout","mac":"000000000000","status":"disconnected"}

只要有此返回，系统需再次下发连接申请。

5、获得数据

【2019-11-18 12:50:32:031】{"FreOrChan":"0","Gateway":"10000015","NodeId":"4F0E58B4","Rssi":"0.0000","Snr":"0.0000","SystemId":"10990143","Type":"ble_evt_disconnected","mac":"4F0E58D4DED4","reason":"62"}
【2019-11-18 12:50:33:836】{"FreOrChan":"0","Gateway":"10000015","NodeId":"4F0E58B4","Rssi":"0.0000","Snr":"0.0000","SystemId":"10990143","Type":"ble_evt_connected","data_length":"64","mac":"4F0E58D4DED4"}
【2019-11-18 12:50:34:166】{"FreOrChan":"0","Gateway":"10000015","NodeId":"4F0E58B4","Rssi":"0.0000","Snr":"0.0000","SystemId":"10990143","Type":"ble_evt_rxd","data":"AA,55,0F,07,02,46,45,43,42,41,FB","mac":"4F0E58D4DED4"}
【2019-11-18 12:50:34:226】{"FreOrChan":"0","Gateway":"10000015","NodeId":"4F0E58B4","Rssi":"0.0000","Snr":"0.0000","SystemId":"10990143","Type":"ble_evt_rxd","data":"AA,55,0F,07,02,40,3F,3E,3D,3C,80","mac":"4F0E58D4DED4"}
【2019-11-18 12:50:34:406】{"FreOrChan":"0","Gateway":"10000015","NodeId":"4F0E58B4","Rssi":"0.0000","Snr":"0.0000","SystemId":"10990143","Type":"ble_evt_rxd","data":"AA,55,0F,07,02,3C,3B,3A,3A,39,0D,AA,55,0F,07,02,39,3B,3B,37","mac":"4F0E58D4DED4"}
【2019-11-18 12:50:34:766】{"FreOrChan":"0","Gateway":"10000015","NodeId":"4F0E58B4","Rssi":"0.0000","Snr":"0.0000","SystemId":"10990143","Type":"ble_evt_rxd","data":"37,0E,AA,55,0F,07,02,36,36,36,36,36,9A","mac":"4F0E58D4DED4"}
【2019-11-18 12:50:34:978】{"FreOrChan":"0","Gateway":"10000015","NodeId":"4F0E58B4","Rssi":"0.0000","Snr":"0.0000","SystemId":"10990143","Type":"ble_evt_rxd","data":"AA,55,0F,07,02,36,36,37,37,37,AB,AA,55,0F,07,02,36,36,36,36","mac":"4F0E58D4DED4"}
【2019-11-18 12:50:35:156】{"FreOrChan":"0","Gateway":"10000015","NodeId":"4F0E58B4","Rssi":"0.0000","Snr":"0.0000","SystemId":"10990143","Type":"ble_evt_rxd","data":"36,78,AA,55,0F,07,02,35,35,35,34,34,95,AA,55,0F,07,02,34,34","mac":"4F0E58D4DED4"}
【2019-11-18 12:50:35:246】{"FreOrChan":"0","Gateway":"10000015","NodeId":"4F0E58B4","Rssi":"0.0000","Snr":"0.0000","SystemId":"10990143","Type":"ble_evt_rxd","data":"34,34,33,FF,AA,55,0F,08,01,00,00,00,00,00,00,96,AA,55,F0,03","mac":"4F0E58D4DED4"}
【2019-11-18 12:50:35:396】{"FreOrChan":"0","Gateway":"10000015","NodeId":"4F0E58B4","Rssi":"0.0000","Snr":"0.0000","SystemId":"10990143","Type":"ble_evt_rxd","data":"03,03,F6,AA,55,0F,06,20,4B,00,26,00,90,AA,55,0F,07,02,33,33","mac":"4F0E58D4DED4"}
【2019-11-18 12:50:35:576】{"FreOrChan":"0","Gateway":"10000015","NodeId":"4F0E58B4","Rssi":"0.0000","Snr":"0.0000","SystemId":"10990143","Type":"ble_evt_rxd","data":"33,33,33,3C,AA,55,0F,07,02,33,33,33,33,33,3C,AA,55,0F,06,21","mac":"4F0E58D4DED4"}
【2019-11-18 12:50:35:636】{"FreOrChan":"0","Gateway":"10000015","NodeId":"4F0E58B4","Rssi":"0.0000","Snr":"0.0000","SystemId":"10990143","Type":"ble_evt_rxd","data":"01,01,00,00,66,AA,55,0F,07,02,33,33,33,33,33,3C,AA,55,0F,07","mac":"4F0E58D4DED4"}
【2019-11-18 12:50:35:786】{"FreOrChan":"0","Gateway":"10000015","NodeId":"4F0E58B4","Rssi":"0.0000","Snr":"0.0000","SystemId":"10990143","Type":"ble_evt_rxd","data":"02,33,33,33,33,33,3C,AA,55,0F,07,02,33,33,34,34,34,AB,AA,55","mac":"4F0E58D4DED4"}
【2019-11-18 12:50:35:996】{"FreOrChan":"0","Gateway":"10000015","NodeId":"4F0E58B4","Rssi":"0.0000","Snr":"0.0000","SystemId":"10990143","Type":"ble_evt_rxd","data":"0F,07,02,34,34,34,B4,33,B0,AA,55,0F,07,02,33,33,33,33,33,3C","mac":"4F0E58D4DED4"}
【2019-11-18 12:50:36:206】{"FreOrChan":"0","Gateway":"10000015","NodeId":"4F0E58B4","Rssi":"0.0000","Snr":"0.0000","SystemId":"10990143","Type":"ble_evt_rxd","data":"AA,55,0F,07,02,33,33,33,33,33,3C,AA,55,0F,07,02,33,31,2E,2B","mac":"4F0E58D4DED4"}
【2019-11-18 12:50:36:596】{"FreOrChan":"0","Gateway":"10000015","NodeId":"4F0E58B4","Rssi":"0.0000","Snr":"0.0000","SystemId":"10990143","Type":"ble_evt_rxd","data":"28,46,AA,55,0F,07,02,24,22,20,1E,1D,7C,AA,55,0F,08,01,00,00","mac":"4F0E58D4DED4"}
【2019-11-18 12:50:36:776】{"FreOrChan":"0","Gateway":"10000015","NodeId":"4F0E58B4","Rssi":"0.0000","Snr":"0.0000","SystemId":"10990143","Type":"ble_evt_rxd","data":"00,00,00,C0,96,AA,55,F0,03,03,03,F6,AA,55,0F,06,20,4B,00,27","mac":"4F0E58D4DED4"}
【2019-11-18 12:50:36:986】{"FreOrChan":"0","Gateway":"10000015","NodeId":"4F0E58B4","Rssi":"0.0000","Snr":"0.0000","SystemId":"10990143","Type":"ble_evt_rxd","data":"00,54,AA,55,0F,07,02,1C,1B,19,18,17,99,AA,55,0F,07,02,17,1A","mac":"4F0E58D4DED4"}
【2019-11-18 12:50:37:000】{"FreOrChan":"0","Gateway":"10000015","NodeId":"00000000","Rssi":"0.0000","Snr":"0.0000","SystemId":"10990143","Type":"station_heart","ip":"192.168.1.132","port":"10604","version":"V-03-01-08"}
【2019-11-18 12:50:37:047】{"FreOrChan":"0","Gateway":"10000015","NodeId":"4F0E58B4","Rssi":"0.0000","Snr":"0.0000","SystemId":"10990143","Type":"ble_evt_rxd","data":"21,2C,39,B7,AA,55,0F,06,21,01,01,00,00,66,AA,55,0F,07,02,47","mac":"4F0E58D4DED4"}
【2019-11-18 12:50:37:226】{"FreOrChan":"0","Gateway":"10000015","NodeId":"4F0E58B4","Rssi":"0.0000","Snr":"0.0000","SystemId":"10990143","Type":"ble_evt_rxd","data":"53,5D,63,66,1C,AA,55,0F,07,02,68,68,66,64,60,FB,AA,55,0F,07","mac":"4F0E58D4DED4"}
【2019-11-18 12:50:37:436】{"FreOrChan":"0","Gateway":"10000015","NodeId":"4F0E58B4","Rssi":"0.0000","Snr":"0.0000","SystemId":"10990143","Type":"ble_evt_rxd","data":"02,5C,58,55,52,D1,F9,AA,55,0F,07,02,4E,4C,49,45,41,52,AA,55","mac":"4F0E58D4DED4"}
【2019-11-18 12:50:37:617】{"FreOrChan":"0","Gateway":"10000015","NodeId":"4F0E58B4","Rssi":"0.0000","Snr":"0.0000","SystemId":"10990143","Type":"ble_evt_rxd","data":"0F,07,02,3C,37,32,2C,28,BF,AA,55,0F,07,02,23,20,1D,1B,19,7A","mac":"4F0E58D4DED4"}
【2019-11-18 12:50:37:796】{"FreOrChan":"0","Gateway":"10000015","NodeId":"4F0E58B4","Rssi":"0.0000","Snr":"0.0000","SystemId":"10990143","Type":"ble_evt_rxd","data":"AA,55,0F,07,02,18,16,14,13,11,F5,AA,55,0F,07,02,0F,10,13,1A","mac":"4F0E58D4DED4"}
【2019-11-18 12:50:38:036】{"FreOrChan":"0","Gateway":"10000015","NodeId":"4F0E58B4","Rssi":"0.0000","Snr":"0.0000","SystemId":"10990143","Type":"ble_evt_rxd","data":"26,24,AA,55,0F,08,01,00,00,00,00,00,96,AA,55,F0,03,03,03","mac":"4F0E58D4DED4"}
【2019-11-18 12:50:38:186】{"FreOrChan":"0","Gateway":"10000015","NodeId":"4F0E58B4","Rssi":"0.0000","Snr":"0.0000","SystemId":"10990143","Type":"ble_evt_rxd","data":"F6,AA,55,0F,06,20,49,AA,55,0F,07,02,5D,5E,5D,5B,D7,77,AA,55","mac":"4F0E58D4DED4"}
【2019-11-18 12:50:38:306】{"FreOrChan":"0","Gateway":"10000015","NodeId":"4F0E58B4","Rssi":"0.0000","Snr":"0.0000","SystemId":"10990143","Type":"ble_evt_rxd","data":"0F,06,21,01,01,00,00,AA,55,0F,07,02,3E,3B,39,37,34,1E,AA,55","mac":"4F0E58D4DED4"}
【2019-11-18 12:50:38:366】{"FreOrChan":"0","Gateway":"10000015","NodeId":"4F0E58B4","Rssi":"0.0000","Snr":"0.0000","SystemId":"10990143","Type":"ble_evt_rxd","data":"0F,07,02,31,2E,2B,28,AA,55,0F,07,02,21,1F,1D,1B,1A,C9,AA,55","mac":"4F0E58D4DED4"}
【2019-11-18 12:50:38:426】{"FreOrChan":"0","Gateway":"10000015","NodeId":"4F0E58B4","Rssi":"0.0000","Snr":"0.0000","SystemId":"10990143","Type":"ble_evt_rxd","data":"0F,07,02,18,17,16,14,13,E7,AA,55,0F,07,02,12,11,12,16,1E,A1","mac":"4F0E58D4DED4"}
【2019-11-18 12:50:38:627】{"FreOrChan":"0","Gateway":"10000015","NodeId":"4F0E58B4","Rssi":"0.0000","Snr":"0.0000","SystemId":"10990143","Type":"ble_evt_rxd","data":"AA,55,0F,07,02,2B,3B,4B,5A,65,46,AA,55,0F,07,02,6C,6F,70,6E","mac":"4F0E58D4DED4"}
【2019-11-18 12:50:38:826】{"FreOrChan":"0","Gateway":"10000015","NodeId":"4F0E58B4","Rssi":"0.0000","Snr":"0.0000","SystemId":"10990143","Type":"ble_evt_rxd","data":"6A,60,AA,55,0F,08,01,AA,55,0F,07,02,47,44,42,40,3D,B6,AA,55","mac":"4F0E58D4DED4"}
【2019-11-18 12:50:39:027】{"FreOrChan":"0","Gateway":"10000015","NodeId":"4F0E58B4","Rssi":"0.0000","Snr":"0.0000","SystemId":"10990143","Type":"ble_evt_rxd","data":"0F,06,21,01,01,00,00,AA,55,0F,07,02,3B,39,37,35,33,2D,AA,55","mac":"4F0E58D4DED4"}
【2019-11-18 12:50:39:226】{"FreOrChan":"0","Gateway":"10000015","NodeId":"4F0E58B4","Rssi":"0.0000","Snr":"0.0000","SystemId":"10990143","Type":"ble_evt_rxd","data":"0F,07,02,31,2F,2E,2C,2C,E7,AA,55,0F,06,21,01,02,1B,00,B2,AA","mac":"4F0E58D4DED4"}
【2019-11-18 12:50:39:426】{"FreOrChan":"0","Gateway":"10000015","NodeId":"4F0E58B4","Rssi":"0.0000","Snr":"0.0000","SystemId":"10990143","Type":"ble_evt_rxd","data":"55,0F,07,02,2B,2B,2B,2B,2B,F0,AA,55,0F,07,02,2B,2D,32,39,43","mac":"4F0E58D4DED4"}
【2019-11-18 12:50:39:626】{"FreOrChan":"0","Gateway":"10000015","NodeId":"4F0E58B4","Rssi":"0.0000","Snr":"0.0000","SystemId":"10990143","Type":"ble_evt_rxd","data":"E7,AA,55,0F,07,02,4D,57,5F,64,67,D0,AA,55,0F,07,02,69,68,65","mac":"4F0E58D4DED4"}
【2019-11-18 12:50:39:826】{"FreOrChan":"0","Gateway":"10000015","NodeId":"4F0E58B4","Rssi":"0.0000","Snr":"0.0000","SystemId":"10990143","Type":"ble_evt_rxd","data":"E0,5A,06,AA,55,0F,07,02,51,4A,46,43,42,5F,AA,55,0F,07,02,43","mac":"4F0E58D4DED4"}
【2019-11-18 12:50:40:026】{"FreOrChan":"0","Gateway":"10000015","NodeId":"4F0E58B4","Rssi":"0.0000","Snr":"0.0000","SystemId":"10990143","Type":"ble_evt_rxd","data":"44,44,46,4C,1A,AA,55,0F,08,01,62,42,00,00,00,03,51,AA,55,F0","mac":"4F0E58D4DED4"}
【2019-11-18 12:50:40:226】{"FreOrChan":"0","Gateway":"10000015","NodeId":"4F0E58B4","Rssi":"0.0000","Snr":"0.0000","SystemId":"10990143","Type":"ble_evt_rxd","data":"03,03,03,F6,AA,55,0F,AA,55,0F,07,02,00,00,00,00,00,28,AA,55","mac":"4F0E58D4DED4"}
【2019-11-18 12:50:40:426】{"FreOrChan":"0","Gateway":"10000015","NodeId":"4F0E58B4","Rssi":"0.0000","Snr":"0.0000","SystemId":"10990143","Type":"ble_evt_rxd","data":"0F,07,02,00,00,00,00,AA,55,0F,07,02,00,00,00,80,00,07,AA,55","mac":"4F0E58D4DED4"}
【2019-11-18 12:50:40:776】{"FreOrChan":"0","Gateway":"10000015","NodeId":"4F0E58B4","Rssi":"0.0000","Snr":"0.0000","SystemId":"10990143","Type":"ble_evt_rxd","data":"0F,06,21,01,02,1D,00,AA,55,0F,07,02,00,00,00,00,00,28,AA,55","mac":"4F0E58D4DED4"}
【2019-11-18 12:50:40:976】{"FreOrChan":"0","Gateway":"10000015","NodeId":"4F0E58B4","Rssi":"0.0000","Snr":"0.0000","SystemId":"10990143","Type":"ble_evt_rxd","data":"0F,07,02,00,00,00,00,AA,55,0F,07,02,00,00,00,00,00,28,AA,55","mac":"4F0E58D4DED4"}
【2019-11-18 12:50:41:176】{"FreOrChan":"0","Gateway":"10000015","NodeId":"4F0E58B4","Rssi":"0.0000","Snr":"0.0000","SystemId":"10990143","Type":"ble_evt_rxd","data":"0F,07,02,00,00,00,00,AA,55,0F,07,02,00,00,00,00,00,28,AA,55","mac":"4F0E58D4DED4"}
【2019-11-18 12:50:41:376】{"FreOrChan":"0","Gateway":"10000015","NodeId":"4F0E58B4","Rssi":"0.0000","Snr":"0.0000","SystemId":"10990143","Type":"ble_evt_rxd","data":"0F,08,01,62,46,00,00,AA,55,0F,07,02,00,00,00,00,00,28,AA,55","mac":"4F0E58D4DED4"}
【2019-11-18 12:50:41:576】{"FreOrChan":"0","Gateway":"10000015","NodeId":"4F0E58B4","Rssi":"0.0000","Snr":"0.0000","SystemId":"10990143","Type":"ble_evt_rxd","data":"0F,07,02,00,00,00,00,AA,55,0F,07,02,00,00,00,00,00,28,AA,55","mac":"4F0E58D4DED4"}
【2019-11-18 12:50:41:776】{"FreOrChan":"0","Gateway":"10000015","NodeId":"4F0E58B4","Rssi":"0.0000","Snr":"0.0000","SystemId":"10990143","Type":"ble_evt_rxd","data":"0F,06,21,01,02,1C,00,AA,55,0F,07,02,00,00,00,00,00,28,AA,55","mac":"4F0E58D4DED4"}
【2019-11-18 12:50:41:976】{"FreOrChan":"0","Gateway":"10000015","NodeId":"4F0E58B4","Rssi":"0.0000","Snr":"0.0000","SystemId":"10990143","Type":"ble_evt_rxd","data":"0F,07,02,00,00,00,00,AA,55,0F,07,02,00,00,00,00,00,28,AA,55","mac":"4F0E58D4DED4"}
【2019-11-18 12:50:42:176】{"FreOrChan":"0","Gateway":"10000015","NodeId":"4F0E58B4","Rssi":"0.0000","Snr":"0.0000","SystemId":"10990143","Type":"ble_evt_rxd","data":"0F,07,02,00,06,12,1D,AA,55,0F,07,02,31,32,32,32,32,01,AA,55","mac":"4F0E58D4DED4"}
【2019-11-18 12:50:42:376】{"FreOrChan":"0","Gateway":"10000015","NodeId":"4F0E58B4","Rssi":"0.0000","Snr":"0.0000","SystemId":"10990143","Type":"ble_evt_rxd","data":"0F,06,01,63,48,00,30,AA,55,0F,06,20,41,02,41,02,4C,AA,55,F0","mac":"4F0E58D4DED4"}
【2019-11-18 12:50:42:576】{"FreOrChan":"0","Gateway":"10000015","NodeId":"4F0E58B4","Rssi":"0.0000","Snr":"0.0000","SystemId":"10990143","Type":"ble_evt_rxd","data":"03,03,03,F6","mac":"4F0E58D4DED4"}

第6章 联网门锁接口

联网门锁接口是采用 ACserver 对接，通过平台对门锁进行控制与管理。

6.1. 接口基础说明

6.1.1. 字典说明

GUID：二代证唯一的卡号，为 8 个字节；
M1 卡：13.56MHz 卡，市面上大部分卡均为 M1 卡；

6.1.2. 接口调用方式

根据不同的操作场景，有不同的操作接口，一般下行数据，对锁或者 ACserver 进行操作，用 http 接口；上行数据，接收实时消息，例如：刷卡上报数据等，用 socket 接口。

6.1.3. 接口调用步骤

平台需要跟 webmain 进行对接进行锁的操作步骤：
1. 在 webmain 里添加基站
2. 在 webmain 里添加一把锁
3. 可以对锁进行添加密码，身份证等操作

6.2. 与 ACserver 数据交互接口

6.2.1. 在 ACserver 添加门锁节点

接口形式：http
接口：http://127.0.0.1:8888/api/lock?cmd=addlock&id={id}&value=card
说明：为 webmain 添加一个锁类型的节点，接口只需要提供节点的 id 即可，例如添加 10000003 的锁节点，就调用接口：
http://127.0.0.1:8888/api/lock?cmd=addlock&id=10000003&value=card
返回：{result：ok}，ok 即执行成功，fail 即执行失败

6.2.2. 在 ACserver 删除门锁节点

接口形式：http
接口：http://127.0.0.1:8888/api/lock?cmd=deletelock&id={id}&value=0
说明：删除锁类型的节点，接口只需要提供节点的 id 即可，例如删除 10000003 的锁节点，就调用接口：http://127.0.0.1:8888/api/lock?cmd=deletelock&id=10000003&value=0
返回：{result：ok}，ok 即执行成功，fail 即执行失败

6.3. 服务器与门锁数据交互接口

服务器与门锁数据交互，是通过服务器、ACserver、网关至门锁的数据交互，所有操作最终均为门锁执行。
注意：与门锁数据交互前需在 ACserver 中添加门锁节点

6.3.1. 远程开门接口

接口形式：http，注意该接口向下访问门锁，返回时间较长约 5s
接口：http://127.0.0.1:8888/api/lock?cmd=ble_unlock_command&id={id}&value=0
说明：打开门锁，接口只需要提供节点的 id 即可，例如打开 10000003 的锁节点，就调用接口
http://127.0.0.1:8888/api/lock?cmd=ble_unlock_command&id=10000003&value=0 即可。
返回：{ running_result：success,result：ok,nodeId:10000003 },
Running_result 的值代表锁的返回结果
success 即执行成功，
decode is wrong 即锁解码失败
pwd is wrong 即锁密钥校验失败
time is wrong 即锁的时间校验失败
ble connection error 即蓝牙连接失败

6.3.2. 同步锁的卡信息

接口形式：http 注意该接口向下访问门锁，返回时间较长约 5s
接口：http://127.0.0.1:8888/api/lock?cmd=ble_syn_command&id={id}&value=0
说明：同步门锁，同步完成后，webmain 将清空该节点的卡表，按门锁上的配置重新配置该门锁的卡配置，接口只需要提供节点的 id 即可，例如同步 10000003 的锁节点，就调用接口
http://127.0.0.1:8888/api/lock?cmd=ble_syn_command&id=10000003&value=0 即可。
返回：


```
{
    "command": "ble_syn_command",
    "data": {
        "battery": 6.503,
        "idcards": [
            {
                "end": "2106-02-07 14:28:15",
                "id": "11CA9002096E9021",
                "start": "1970-01-01 08:00:00"
            }
        ],
        "pwds": [
            {
```

```
                    "end": "2106-02-07 14:28:15",
                    "pwd": "24680200",
                    "start": "1970-01-01 08:00:00"
                },
                {
                    "end": "2106-02-07 14:28:15",
                    "pwd": "13579000",
                    "start": "1970-01-01 08:00:00"
                }
            ],
            "status": "normal",
            "version": "00-3C-1B"
        },
        "nodeId": "10000092",
        "result": "ok",
        "running_result": "success"
}
```


Data：里面是锁同步回来的数据，battery 是电量；idcards 是身份证号信息；pwds 是密码信息等

Running_result 的值代表锁的返回结果

success 即执行成功，

decode is wrong 即锁解码失败

pwd is wrong 即锁密钥校验失败

time is wrong 即锁的时间校验失败

ble connection error 即蓝牙连接失败

6.3.3. 同步锁离线消息

接口形式：http 注意该接口向下访问门锁，返回时间较长约 5s

接口：http://127.0.0.1:8888/api/lock?cmd=ble_synlog_command&id={id}&value=0

说明：同步门锁，同步完成后，webmain 将清空该节点的卡表，按门锁上的配置重新配置该门锁的卡配置，接口只需要提供节点的 id 即可，例如同步 10000003 的锁节点，就调用接口 http://127.0.0.1:8888/api/lock?cmd=ble_synlog_command&id=10000003&value=0 即可。

返回：


```
{
    "command": "ble_synlog_command",
    "data": [
        {
            "CommandType": "card",
            "cardNo": "11CA9002096E9021",
            "cardType": "BID",
            "failReason": 0,
            "openDoor": "fail",
```

```
            "time": "2020-01-08 18:12:30"
        },
        {
            "CommandType": "card",
            "cardNo": "11CA9002096E9021",
            "cardType": "BID",
            "openDoor": "success",
            "time": "2020-01-08 18:12:35"
        }
    ],
    "nodeId": "10000092",
    "result": "ok",
    "running_result": "success"
}
```


Data：里面是锁同步回来的数据，commandType 如果是 card 就是卡开门的记录，如果是 pwd，就是密码开门的记录

Running_result 的值代表锁的返回结果

success 即执行成功，

decode is wrong 即锁解码失败

pwd is wrong 即锁密钥校验失败

time is wrong 即锁的时间校验失败

ble connection error 即蓝牙连接失败

6.3.4. 添加门锁卡（M1 或 GUID）授权

接口形式：http 注意该接口向下访问门锁，返回时间较长约 5s

接口：

http://127.0.0.1:8888/api/lock?cmd=ble_addcard_command&id={id}&name={name}&cardno={cardno}&start={timestart}&end={timeend}&type={cardtype}&offtype=offline&total=1000

说明：添加一张卡到锁上，

接口参数：

Id：节点的 id

Name：给添加卡起的名称

Cardno：卡号，如果是 M1 卡就是 8 位 4 字节的号，如果是身份证，就是 16 位 8 字节的号，比如，cardno= F406CB26 卡号就是 0xf4,0x06,0xcb,0x26

Start:添加的卡号生效的开始时间，格式是 yyyy-MM-dd_HH:mm:ss，例如：2019-10-10_07:00:00

end:添加的卡号生效的结束时间，格式是 yyyy-MM-dd_HH:mm:ss，例如：2019-10-10_07:00:00

type：添加卡号的类型，如果是 M1 卡就是 AM1，如果是身份证就是 BID

offtype:固定为 offline

total：固定为 1000

例如
http://127.0.0.1:8888/api/lock?cmd=ble_addcard_command&id=10000003&name=card1&cardno=F406CB26&start=2019-10-10_07:00:00&end=2019-11-10_07:00:00&type=AM1&offtype=offline&total=1000
返回：{ running_result：success,result：ok,nodeId:10000003 },
Running_result 的值代表锁的返回结果
success 即执行成功,
decode is wrong 即锁解码失败
pwd is wrong 即锁密钥校验失败
time is wrong 即锁的时间校验失败
ble connection error 即蓝牙连接失败
lock m1 card already in lock m1 卡号已经在锁里存在
lock id card already in lock 身份证卡号已经在锁离存在
lock card enough 锁的存储空间已经满了

6.3.5. 删除门锁卡（M1 或 GUID）授权

接口形式：http 注意该接口向下访问门锁，返回时间较长约 5s
接口：
http://127.0.0.1:8888/api/lock?cmd=ble_deletecard_command&id={id}&value=0
说明：从门锁删除一张卡，接口只需要提供卡的表 id 即可（注意是表 id 不是卡号），例如删除表 id 为 10 的卡，就调用接口
http://127.0.0.1:8888/api/lock?cmd=ble_deletecard_command&id=10&value=0 即可。
返回：{ running_result：success,result：ok,nodeId:10000003 },
Running_result 的值代表锁的返回结果
success 即执行成功,
decode is wrong 即锁解码失败
pwd is wrong 即锁密钥校验失败
time is wrong 即锁的时间校验失败
ble connection error 即蓝牙连接失败
lock card not in lock 卡号没有在锁里

6.3.6. 获取门锁基础信息

接口形式：http
接口：
http://127.0.0.1:8888/api/datalist?cmd=lock_node
该接口返回一个 json 数组，数组里面是每个锁的信息，数组里的对象字段解析如下：
NodeId:锁的节点 id
LockType：锁的类型，固定为 card
Battery：锁的电池电压值
Status：锁的状态
TimeBase：锁的时间参考值，可忽略

AskTimeInterval：锁的周期唤醒时间，蓝牙锁忽略
FirewareVersion：固件版本号
ConfigVersion：配置版本号，蓝牙锁忽略

例如：


```
[
 {
     "NodeId": "10000004",
     "LockType": "10000004",
     "Status": "normal",
     "Battery": 7,
     "Timebase": 0,
     "AskTimeInterval": 3500000,
     "FirewareVersion": "00-00-24",
     "ConfigVersion": 0,
     "CurrentCardNo": ""
 },
 {
     "NodeId": "10000005",
     "LockType": "10000005",
     "Status": "normal",
     "Battery": 2355,
     "Timebase": 0,
     "AskTimeInterval": 3500000,
     "FirewareVersion": "00-00-24",
     "ConfigVersion": 0,
     "CurrentCardNo": ""
 },
 {
     "NodeId": "10000006",
     "LockType": "card",
     "Status": "normal",
     "Battery": 28,
     "Timebase": 0,
     "AskTimeInterval": 3500000,
     "FirewareVersion": "00-00-24",
     "ConfigVersion": 0,
     "CurrentCardNo": ""
 },
```

```
 {
     "NodeId": "10000003",
     "LockType": "10000003",
     "Status": "normal",
     "Battery": 0,
     "Timebase": 0,
     "AskTimeInterval": 3500000,
     "FirewareVersion": "00-00-24",
     "ConfigVersion": 0,
     "CurrentCardNo": ""
 }
]
```


6.3.7. 取得所有门锁卡信息

接口形式：http
接口：
http://127.0.0.1:8888/api/datalist?cmd=lock_card
该接口返回一个 json 数组，数组里面是每个卡的信息，数组里的对象字段解析如下：
Id：卡的表 id，删除卡的时候能用上
NodeId：这张卡在那个锁节点上，这个是节点 id
CardName：设置的卡名称
CardNo：卡号
CardType：卡类型，如果是 m1 卡就是 AM1，如果是身份证就是 BID
StartTime：卡号生效的开始时间，格式是 yyyy-MM-dd HH:mm:ss，例如：2019-10-10 07:00:00
EndTime：卡号生效的结束时间，格式是 yyyy-MM-dd HH:mm:ss，例如：2019-10-10 07:00:00
OfflineType：固定为 offline
TotalTimes：忽略
UseTimes：忽略
例如：


```
[

 {
     "CardName": "F406CB26",
     "CardNo": "F406CB26",
     "CardType": "AM1",
     "EndTime": "2019-12-01 00:00:00",
     "Id": 68,
```

```
        "NodeId": "10000003",
        "OfflineType": "offline",
        "StartTime": "2019-01-01 00:00:00",
        "TotalTimes": 10000,
        "UseTimes": 0
    }
]
```

6.3.8. 门锁日志上报

接口形式：socket
刷卡开门成功，上报信息示例：

```
{
  "FreOrChan": "0",
  "Gateway": "10000017",
  "NodeId": "10000003",
  "Rssi": "0.0000",
  "Snr": "0.0000",
  "SystemId": "10990143",
  "Type": "ble_lock_card",
  "cardNo": "F406CB26",
  "cardType": "AM1",
  "openResult": "success"
}
```

字段说明
FreOrChan：忽略
Rssi：忽略
Snr：忽略
SystemId：webmain 的 id 号
Gateway：接收基站的 id
NodeId：锁的节点 id
Type：固定为 ble_lock_card，为刷卡开锁的日志上报信息
cardNo：卡号
cardType：卡类型
openResult：刷卡结果成功

刷卡开门失败，上报信息示例：

```
{
  "FreOrChan": "0",
  "Gateway": "10000017",
```

```
    "NodeId": "10000003",
    "Rssi": "0.0000",
    "Snr": "0.0000",
    "SystemId": "10990143",
    "Type": "ble_lock_card",
    "cardNo": "F406CB26",
    "cardType": "AM1",
    "openResult": "success",
    "failReason": "nocard"
  }
```

字段说明
FreOrChan：忽略
Rssi：忽略
Snr：忽略
SystemId：webmain 的 id 号
Gateway：接收基站的 id
NodeId：锁的节点 id
Type：固定为 ble_lock_card，为刷卡开锁的日志上报信息
cardNo：卡号
cardType：卡类型
openResult：刷卡结果失败
failReason：失败原因，如果是 nocard 就是没有卡号配置在锁里，如果是 timewrong，时间错误。

6.3.9. 门锁电量上报

接口形式：socket
门锁周期上报电池电量信息，上报信息示例：

```
{
  "FreOrChan": "0",
  "Gateway": "10000019",
  "NodeId": "00000000",
  "Rssi": "0.0000",
  "Snr": "0.0000",
  "SystemId": "10990143",
  "Type": "ble_station_online",

  "online": [

      {
          "battery": 6.503,
          "nodeId": "10000092",
```

```
            "nodeType": "lock",
            "offlineItem": 0,
            "rssi": -52,
            "version": 2
        }
    ]
}
```

字段说明
FreOrChan：忽略
Rssi：忽略
Snr：忽略
SystemId：webmain 的 id 号
Gateway：接收基站的 id
NodeId：锁的节点 id
Type：固定为 ble_lock_log，为门锁周期上报日志信息
battery：电池电压，如果 4582，就是 4.582v

6.3.10. 二代证卡面号认证刷卡

说明：二代证卡面号授权是在平台预定时输入入住人的姓名、身份证号等信息，到门前可直接刷卡开门。此业务流程仅适用与短租、网约房等场景，若需要请与金桔联系确认再行开发。
接口形式：socket
当门锁操作进行用二代证卡面号认证时，主动上报的信息。

```
{
 "FreOrChan": "0",
 "Gateway": "10000019",
 "NodeId": "10000092",
 "Rssi": "0.0000",
 "Snr": "0.0000",
 "SystemId": "10990143",
 "Type": "ble_lock_card_record",
 "cardNo": "11CA9002096E8020",
 "cardType": "BID",
 "end": "2106-02-07 14:28:15",
 "pwd": "13579001",
 "recordResult": "success",
 "start": "1970-01-01 08:00:00"
```

6.3.11. 添加密码接口

接口形式：http 注意该接口向下访问门锁，返回时间较长约 5s

接口：
http://127.0.0.1:8888/api/lock?cmd=ble_addpwd_command&id={id}&name={name}&pwd={pwd}&start={timestart}&end={timeend}&total=1000
说明：添加一个密码到锁上，
接口参数：
Id：节点的 id
Name：给添加密码的名称
pwd：密码，可以用的密码一共 6 位，这里占用 8 个字节（主要为了 4 字节对齐），后两个字节为 00，比如设定的密码为 123456，就传入 12345600；如果设定的密码为 112233，就传入 11223300
Start:添加的密码生效的开始时间，格式是 yyyy-MM-dd_HH:mm:ss，例如：2019-10-10_07:00:00
end:添加的密码生效的结束时间，格式是 yyyy-MM-dd_HH:mm:ss，例如：2019-10-10_07:00:00
total：固定为 1000
例如
http://127.0.0.1:8888/api/lock?cmd=ble_addpwd_command&id=10000003&name=pwd1&pwd=12345600&start=2019-10-10_07:00:00&end=2019-11-10_07:00:00&total=1000
返回：{ running_result：success,result：ok,nodeId:10000003 },
Running_result 的值代表锁的返回结果
success 即执行成功，
decode is wrong 即锁解码失败
pwd is wrong 即锁密钥校验失败
time is wrong 即锁的时间校验失败
ble connection error 即蓝牙连接失败
lock pwd already in lock 密码已经在锁里存在
lock pwd enough 密码的存储空间已经满了

6.3.12. 删除密码接口

接口形式：http 注意该接口向下访问门锁，返回时间较长约 5s
接口：
http://127.0.0.1:8888/api/lock?cmd=ble_deletecard_command&id={id}&value=0
说明：从门锁删除一张卡，接口只需要提供密码的表 id 即可（注意是表 id 不是密码），例如删除表 id 为 10 的密码，就调用接口
http://127.0.0.1:8888/api/lock?cmd=ble_deletepwd_command&id=10&value=0 即可。
返回：{ running_result：success,result：ok,nodeId:10000003 },
Running_result 的值代表锁的返回结果
success 即执行成功，
decode is wrong 即锁解码失败
pwd is wrong 即锁密钥校验失败
time is wrong 即锁的时间校验失败
ble connection error 即蓝牙连接失败

lock pwd not in lock　密码没有在锁里

6.3.13.　读取门锁当前密码信息

接口形式：http
接口：
http://127.0.0.1:8888/api/datalist?cmd=lock_pwd
该接口返回一个 json 数组，数组里面是每个密码的信息，数组里的对象字段解析如下：
Id：密码的表 id，删除密码的时候能用上
NodeId：这个密码在那个锁节点上，这个是节点 id
Name：设置的密码名称
Pwd：密码，注意这个是带着后面 00 的，比如：密码是 123456，该字段就是 12345600，如果密码是 112233，该字段就是 11223300
StartTime：密码生效的开始时间，格式是 yyyy-MM-dd HH:mm:ss，例如：2019-10-10 07:00:00
EndTime：密码生效的结束时间，格式是 yyyy-MM-dd HH:mm:ss，例如：2019-10-10 07:00:00
TotalTimes：忽略
UseTimes：忽略
例如：


```
[

 {
     "Name": "pwd1",
     "Pwd": "12345600",
     "EndTime": "2019-12-01 00:00:00",
     "Id": 68,
     "NodeId": "10000003",
     "StartTime": "2019-01-01 00:00:00",
     "TotalTimes": 10000,
     "UseTimes": 0
 }
]
```


6.4. 蓝牙设备与门锁数据交互接口

蓝牙设备（手机等）与门锁数据交互主要是。

6.4.1. 微信小程序与门锁交互源码工程

微信小程序的交互示例为开源工程，客户可自行在此基础上开发，我方也欢迎一起维护此开源工程。

源码地址：
https://github.com/jinguolock/wxlock_demo

6.4.2. 微信小程序与门锁交互示例

6.4.2.1. 小程序连接门锁示例

```
connect() {
    if (!wx.openBluetoothAdapter) {
      this.showError("当前微信版本过低，无法使用该功能，请升级到最新微信版本后重试。");
      return;
    }

    var _this = this;
    wx.openBluetoothAdapter({
      success: function (res) {
        console.log("ble ready complete")
        _this.startSearch();
       }
    })
  },
```

connect 连接以后，在 success 里面调用了 startSearch 方法，该方法是查询周边范围内可以连接的所有蓝牙设备。

```
startSearch() {
    var _this = this;
    wx.startBluetoothDevicesDiscovery({
      services: [],
      success(res) {
        wx.onBluetoothDeviceFound(function (res) {
          //console.log("device length:" + res.devices[0].name )
          console.log("device length:" + res.devices.length+"device name:" + res.devices[0].name + ";device mac:" + res.devices[0].deviceId)
          //var device = _this.filterDevice(res.devices);

          if (res.devices[0].name == _this.blue_data.device_id) {
            _this.blue_data.device_id = res.devices[0].deviceId;
```

```
                _this.connectDevice();
                _this.stopSearch();

            }
          });
        }
      })
  },
```

这里面我们自己定义了需要连接的 deviceid 当 res.devices[0].name == _this.blue_data.device_id 查询到时，我们将停止搜索，并试图连接该蓝牙设备（金桔智能锁）。

```
connectDevice() {
    var _this = this;
    console.log("connectDevice device id:" + _this.blue_data.device_id)
    wx.createBLEConnection({
      deviceId: _this.blue_data.device_id,
      success(res) {
        console.log("connectDevice success:" + _this.blue_data.device_id)
        _this.getDeviceService();


      }
    })
  },
```

连接到设备成功以后需要取得该设备的服务列表，因为我们是用的 nus 服务，所以服务 id 在程序中默认定义好了

```
var nus_service = "6E400001-B5A3-F393-E0A9-E50E24DCCA9E"
```

取得服务的方法

```
getDeviceService() {
    var _this = this;
    wx.getBLEDeviceServices({
      deviceId: _this.blue_data.device_id,
      success: function (res) {
        for (var i = 0; i < res.services.length;i++){
          var service_id = res.services[i].uuid
          if (service_id == nus_service) {
            _this.getDeviceCharacter()
          }
        }

      }
    })
  },
```

当我们查到到该服务以后查找接收和发送的特征值，这两个特征值也是事先定义好的：

```
var char_tx = "6E400002-B5A3-F393-E0A9-E50E24DCCA9E"
```

```
var char_rx = "6E400003-B5A3-F393-E0A9-E50E24DCCA9E"
```

取得服务特征值的方法：

```
getDeviceCharacter() {
    let _this = this;
    wx.getBLEDeviceCharacteristics({
      deviceId: _this.blue_data.device_id,
      serviceId: _this.blue_data.service_id,
      success: function (res) {
        /*
        let notify_id, write_id, read_id;
        for (let i = 0; i < res.characteristics.length; i++) {
          let charc = res.characteristics[i];
          if (charc.properties.notify) {
            notify_id = charc.uuid;
          }
          if (charc.properties.write) {
            write_id = charc.uuid;
          }
          if (charc.properties.write) {
            read_id = charc.uuid;
          }
        }
        if (notify_id != null && write_id != null) {
          _this.blue_data.notify_id = notify_id;
          _this.blue_data.write_id = write_id;
          _this.blue_data.read_id = read_id;


        }*/

        _this.openNotify();
      }
    })
  },
```

需要搜索的信息都完成后，通知程序可以发送数据和接收数了，这里的 openNotify 方法就是绑定接收监听到微信小程序里。

```
openNotify() {
    var _this = this;
    wx.notifyBLECharacteristicValueChange({
      state: true,
      deviceId: _this.blue_data.device_id,
      serviceId: _this.blue_data.service_id,
      characteristicId: _this.blue_data.notify_id,
      complete(res) {
```

```
            console.log("notify enable")
            _this.onNotifyChange()
            _this.onOpenNotifyListener && _this.onOpenNotifyListener();

        }
      })
  },
```

可以看到这里当收到数据的时候，我们调用了 onNotifyChange 方法和 onOpenNotifyListener 方法，这两个方法是自己定义的，用户可以自己定义这些方法来接收数据。
比如，示例中是将取得的数据放入到了缓存中

```
  onNotifyChange() {
    var _this = this;
    wx.onBLECharacteristicValueChange(function (res) {
      _this.addToCache(res.value);
    })
  },
```

当需要发送数据时，这里封装了一个发送 16 进制数组的方法，

```
sendHex(bs) {
    let _this = this;
    var buf = new ArrayBuffer(bs.length);
    let dataView = new DataView(buf)
    for (var i = 0, len = bs.length; i < len; i ++) {
      dataView.setUint8(i,bs[i])
    }
    console.log("send hex:" + _this.arrayBufferToHexString(buf))
    wx.writeBLECharacteristicValue({
      deviceId: _this.blue_data.device_id,
      serviceId: _this.blue_data.service_id,
      characteristicId: _this.blue_data.write_id,
      value: buf,
      success: function (res) {
       // console.log(res);
      }
    })
  },
```

6.4.2.2.小程序数据交互与响应示例

使用微信小程序开发智能锁应用时，需要通过 http 服务来获取响应的数据和加密信息。

```
mymain(requestName,cmd, paramObj, callback) {
    var _this = this;
    if(paramObj==null){
      paramObj=new Object();
```

```
    }
    paramObj["mymain_name"] = requestName;
    paramObj["cmd"] = cmd;
    paramObj["c"] = "mymainRequest";
    wx.request({
      url: _this.server.url,
      data: paramObj,
      success: function (res) {
        console.log(res)
        callback && callback(res.data);
      }
    })
  },
```

这里封装了访问方法，调用 wx.request 即可调用 http 服务。

调用时这样使用：

```
sendBleByAuth(deviceId, content, command, reFunc, msgFunc, sendFinishFunc,failFunc) {
    var _this = this;
    var willsendbyte;
    myApi.mymain("lock", "getlockpwd", { id: deviceId }, function (obj) {
      if (obj != null && obj.pwd != null) {
        msgFunc && msgFunc("数据完成，准备操作……")
        willsendbyte = _this.getHexByStr(obj.pwd);
        blueApi.simpleSendBleMsg("IR" + deviceId, content, willsendbyte, command, 30000,
true, reFunc, msgFunc, sendFinishFunc,failFunc)
      }
    }, function (err) {
      msgFunc && msgFunc("准备数据失败，网络错误:"+err)
    }
    )
  },
```

6.4.2.3.小程序数据交互数据格式示例

使用微信小程序开发智能锁应用时，需要通过 http 服务来获取响应的数据和加密信息，http 获取的是字符串，但是智能锁处理的是十六进制数组，所以经常会用到两者的相互转换问题。

hex->string：转换完的字符串是逗号分开的 16 进制，比如：00,0A,02

```
 getStrByHex(hexArr){
    var hexStr = '';
    for (var i = 0; i < hexArr.length; i++) {
      var str = hexArr[i];
      var hex = (str & 0xff).toString(16);
      hex = (hex.length === 1) ? '0' + hex : hex;
      hexStr += hex;
```

```
        if (i!=(hexArr.length-1)){
            hexStr+=',';
        }
    }

    return hexStr;
  },
hex->string：转换完的字符串是在一起的 16 进制，比如：000A02
getStrByHex2(hexArr) {
    var hexStr = '';
    for (var i = 0; i < hexArr.length; i++) {
        var str = hexArr[i];
        var hex = (str & 0xff).toString(16);
        hex = (hex.length === 1) ? '0' + hex : hex;
        hexStr += hex;
    }
    return hexStr;
  },
string->hex：需要转换的字符串是逗号分开的，比如：00,0A,02
getHexByStr(hexStr) {
    var strs = hexStr.split(',');
    var re = new Uint8Array(strs.length);
    for (var i = 0; i < strs.length; i++) {
        re[i] = parseInt(strs[i], 16) & 0xff;
    }

    return re;
  },
string->hex：需要转换的字符串是在一起的，比如：000A02
 getHexByStr2(hexStr) {
    if (!hexStr) {
        return new Uint8Array(0);
    }
    var content = new Uint8Array(hexStr.length/2);
    let ind = 0;
    for (var i = 0, len = hexStr.length; i < len; i += 2) {
        let code = parseInt(hexStr.substr(i, 2), 16)
        content[ind]=code;
        ind++
    }
    return content;
  },
```

6.4.2.4.小程序数据交互时间格式示例

使用微信小程序开发智能锁应用时，需要处理时间格式问题。
date->string:这里的时间格式是，yyyy-MM-dd HH:mm:ss，使用时，new Date().formatTime

```
const formatTime = date => {
  const year = date.getFullYear()
  const month = date.getMonth() + 1
  const day = date.getDate()
  const hour = date.getHours()
  const minute = date.getMinutes()
  const second = date.getSeconds()

  return [year, month, day].map(formatNumber).join('-') + ' ' + [hour, minute, second].map(formatNumber).join(':')
}
```

string->date:这里的时间格式为 yyyy-MM-dd_HH:mm:ss，注意中间的下划线，因为在 http 提交的时候，空格容易造成影响，所以加上下划线。

```
getDateByStr(dateStr) {
      var strs = dateStr.split("_");
      var d1 = strs[0].split("-");
      var d2 = strs[1].split(":");
      var re=new Date();
      re.setFullYear(parseInt(d1[0], 10), parseInt(d1[1], 10) - 1, parseInt(d1[2], 10));
      re.setHours(parseInt(d2[0], 10), parseInt(d2[1], 10), parseInt(d2[2], 10));
      return re
  }
```