# 一、获取模块：

```
var leidenSDK = api.require('leidenSDK');
```

# 二、扫描接口：

```
leidenSDK.scan(function(ret,err) {
    console.log(JSON.stringify(ret));
})
```

1、调用此接口，要确保当前 App 已经申请了位置权限。如果没有申请，此接口会进行申请，但不会有任何返回值和回调。

2、回调会重复回调

扫描到设备的回调数据为：

```
{"type":"devices","name":"LG920BW","mac":"DC:0D:30:64:FB:D6"}
```

此接口会扫描蓝牙 10 秒，结束时的回调数据为：

```
{"type":"end"}
```

# 三、连接接口：

```
var json = {"mac":"DC:0D:30:64:FB:D6"};
leidenSDK.connect(json,function(ret,err) {
    console.log(JSON.stringify(ret));
})
```

1、mac 参数为蓝牙设备的 mac。可以是通过 sacan 接口扫描到的设备的 mac 参数。也可以是已配对的设备的 mac（一般而言，连接过的设备即为已配对，除非用户手动在系统设置处取消了配对）

2、返回的数据格式为：

```
{"name":"LG920BW","mac":"DC:0D:30:64:FB:D6","result":"success"}
```

result 参数代表成功或者失败。

# 四、打印接口

```
function printf() {
```

```
var json = {
        "labels":[
            {
                "labelW":50 * 8,//标签的宽度 50 的单位 mm，8 为打印机上 mm 与 px 的转化单位。必须
                "labelH":50 * 8,//同标签宽度。必须
                "number":1,// 打印的张数。不必须 默认为 1
                "bitmaps":[// 打印的图片数据。不必须。没有数据则打印一张空白标签
                    {
                        "x":0 * 8,//同标签宽度。当前图片的 x 坐标。必须
                        "y":0 * 8,//同标签宽度。当前图片的 x 坐标。必须
                        "WD":25 * 8,//同标签宽度。当前图片的宽度。必须
                        "HT":25 * 8,//同标签宽度。当前图片的高度。必须
                        "bitmapPath":"/storage/emulated/0/logo.png"// 图片路径。必须
                    }
                ]
            },{
                "labelW":50 * 8,//标签的宽度 50 的单位 mm，8 为打印机上 mm 与 px 的转化单位。必须
                "labelH":100 * 8,//同标签宽度。必须
                "number":2,// 打印的张数。不必须 默认为 1
                "bitmaps":[// 打印的图片数据。不必须。没有数据则打印一张空白标签
                    {
                        "x":0 * 8,//同标签宽度。当前图片的 x 坐标。必须
                        "y":0 * 8,//同标签宽度。当前图片的 x 坐标。必须
                        "WD":30 * 8,//同标签宽度。当前图片的宽度。必须
                        "HT":30 * 8,//同标签宽度。当前图片的高度。必须
                        "bitmapPath":"/storage/emulated/0/logo.png"// 图片路径必须
                    }
                ]
            }
        ]
    };
    leidenSDK.printfLabels(json,function(ret,err) {
        console.log(JSON.stringify(ret));
    })
}
```

1、主要注意的是：bitmapPath 要保证有图片并且有权限访问(Android10 无法访问沙盒外的图片，切记)

返回的数据：

{"resultType":1,"result":"打印成功"}
resultType 代表的意义：
//打印成功
int LEIDEN_PRINTF_RESULT_SUCCESS = 1;
//打印错误
int LEIDEN_PRINTF_RESULT_CMD_ERROR = 2;
//蓝牙未连接
int LEIDEN_PRINTF_RESULT_BLUETOOTH = 3;

# 五、环境变量

1、一般不要在开发时调整好 JSON，不要开放给用户设置
2、环境变量设置接口，不需要进行蓝牙连接
3、环境变量的 JSON 代表的意义：
//打印速度　1 -- 高速　2 -- 标准 3 -- 中速 4 -- 低速
private int printfSpeed = 2;
//打印浓度　范围 1 - 15
private int printfPotency = 8;
//打印媒介　1 - 热敏　　2 - 碳带
private int printfMedium = 2;
//标签种类　1 - 黑标　　2 - 模切　　3 - 连续
private int labelType = 3;
//顶部偏移　±支持标签最大高度　-10 --- +10
private int topDeviation = 0;
//打印模式　1 - 标准模式　2 - 连续模式 3 - 剥离模式 4 - 切刀模式
private int printfModel = 1;
//剥离方式 1 - 传感器 2 - 按键 当 printfMode = 剥离模式时有效
private int beStrippedModel = 1;
//剥离送纸量 当 printfMode = 剥离模式时有效
private int beStrippedFeedVolume = 0;
//切纸张数 [0-9999] 当 printfModel == 切刀模式模式时有效
private int cutNumber = 0;
//切刀送纸量 当 printfModel == 切刀模式时有效
private int cutterFeedVolume = 0;
//标准送纸量 当 printfModel == 标准模式时有效
private int standardFeedVolume = 0;

5、设置的为连续纸的案例：
```
function setCurrentEnvContinue() {
    leidenSDK.getCurrentEnviron(function(ret,err){
        console.log(JSON.stringify(ret));
```

```
            ret.labelType = 3;
            leidenSDK.setCurrentEnviron(ret,function(ret,err){
                alert(JSON.stringify(ret));
            });
        })
}
```

# 6、设置为标签纸的案例：

```
function setCurrentEnvLabel() {
        leidenSDK.getCurrentEnviron(function(ret,err){
            ret.labelType = 2;
            leidenSDK.setCurrentEnviron(ret,function(ret,err){
                alert(JSON.stringify(ret));
            });
        })
}
```

# 六、当前是否已连接

调用案例：

```
leidenSDK.isConnect(function(ret,err) {
        alert(JSON.stringify(ret));
});
```

返回的数据：

{“result”:true/false}

七、直接写指令

调用案例：

```
var json = {"content":"JOB\nDEF PW=560,PH=640\nSTART\nQTY P=1\nEND\nJOBE\n"};
leidenSDK.writer(json,function(ret,err) {
        if (ret.result == 1) {
            alert("发送成功");
        } else {
            alert("发送失败");
        }
})
```

# 八、关闭连接

调用案例：

```
leidenSDK.closeConnect();
```