IOS端SDK 接口说明文档 V4.0.3

文档修改记录

序号	版本号	修改内容	修改者	修改日期
1	v1.0.1	文档建立	王亚东	2019/3/9
2	v2.0.0	1.新增D11系列打机支持2.新增B21系列打印机支持3.新增B3S系列打印机支持4.新增P1系列打印机支持	虞明义/张彬	2020/2/4
3	v2.0.1	固件升级优化，条码优化	虞明义	2020/3/30
4	V2.1.1	B21 wifi功能，p1打印机	虞明义	2020/4/21
5	v2.1.2	1、p1增加总张数指令2、p1增加空白页打印3、p1增加打印完之后碳带使用回调	虞明义	2020/5/8
6	v2.1.3	1、适配b162、错误码统一3、p1面板升级以及细节优化	虞明义	2020/7/7
7	v3.0.0	1、新增B50系列打印机支持2、新增B3系列打印机支持3、图像二期接入	虞明义	2020/7/20
8	v3.0.1	1、新增b3s 4.0.1适配2、新增b16适配3、新增不支持指令的适配4、打印第二个参数变更	虞明义	2020/9/1
9	v3.1.0		虞明义	2020/9/14

		1、新增适配B32机型2、新增日志功能3、新增打印缓存，最多5页数据		
10	v3.1.1	1、适配D1102、新增适配8761蓝牙芯片，需要开启	虞明义	2021/3/25
11	V3.1.2	1、去掉适配8761蓝牙接口，默认兼容所有蓝牙接入方式2、新增第三方调用图像库绘制功能接口	虞明义	2021/3/25
12	v3.1.3	1、适配S6机型2、适配300dpi精度修改为11.813、向下同步3.1.2_beta6新增功能4、适配新协议，优化打印超时问题5、修改新协议机型退至后台打印时，时间间隔为1ms6、适配Z401/Z401R7、D101裁切改居中裁切8、回退图像库2.0.10_beta4	虞明义	2021/8/23
13	v3.1.4	1、适配私有协议V3 版本，支持边打印边发送数据2、适配兆讯b3s3、适配B18	虞明义	2021/9/26
14	V3.1.5	1、适配B203机型2、新增b203发送数据间隔控制指令3、第三方机器适配打印bitmap二值化数据	虞明义	2021/11/10
15	V3.1.6	1.适配B3S及B32新机型	张彬	2022/7/15

16	v3.1.7	1、适配p1s新协议打印2、新增适配B32/A63新协议机型，黑标纸上下各裁切1mm3、适配h1s虚线设置	虞明义	2022/8/1
17	v3.1.8	1、适配双色打印协议	虞明义	2022/11/1
18	v3.1.9	1.适配K3、K3_W新机型2.增加WIFI相关接口3.移除废弃的接口	虞明义	2023/8/7
19	v3.2.0	1.移除废弃的接口	虞明义	2023/10/24
20	v3.2.0	1.增加设置SDK缓存接口以解决批量打印时提交任务成功后回调异常问题	张彬	2024/1/3
21	v3.2.4	功能新增:支持M2机型支持B21_H机型支持B3S_P、A8_P、S6_P机型支持B31机型更换新图像库 b. 完善老图像库部分不生效的功能 c. 修复老图像库部分BUG支持部分机型获取打印机内部安装的标签尺寸（M2）	虞明义	2024/7/19
22	v3.2.8	功能新增:支持B21 Pro机型支持K2机型BUG修复:文本换行模式6与其他端不一致问题	虞明义	2025/3/15
23	V4.0.3	功能新增:支持M3机型支持B4机型功能优化:优化接入SDK引入文件	虞明义	2025/6/30

24	V4.0.4	BUG修复:修复B3S打印不清晰的问题	虞明义	2025/8/26

产品目的

JCAPI 接口方法说明文档，是针对标签绘图提出的接口方法说明，方便用

户在二次开发中调用接口，缩短开发周期，加快开发进度。

产品功能

JCAPI 接口为调用者提供易用的方法完成标签绘图的操作。本接口中提供了文字、一维条码、二维码、图片、和各种图形等多种对象的绘制方法，同时还能进行绘制对象的旋转，调用者还可以调用方法获得绘制完成的标签图片用于标签预览,使得标签绘图的操作更加便捷。

一、工程配置

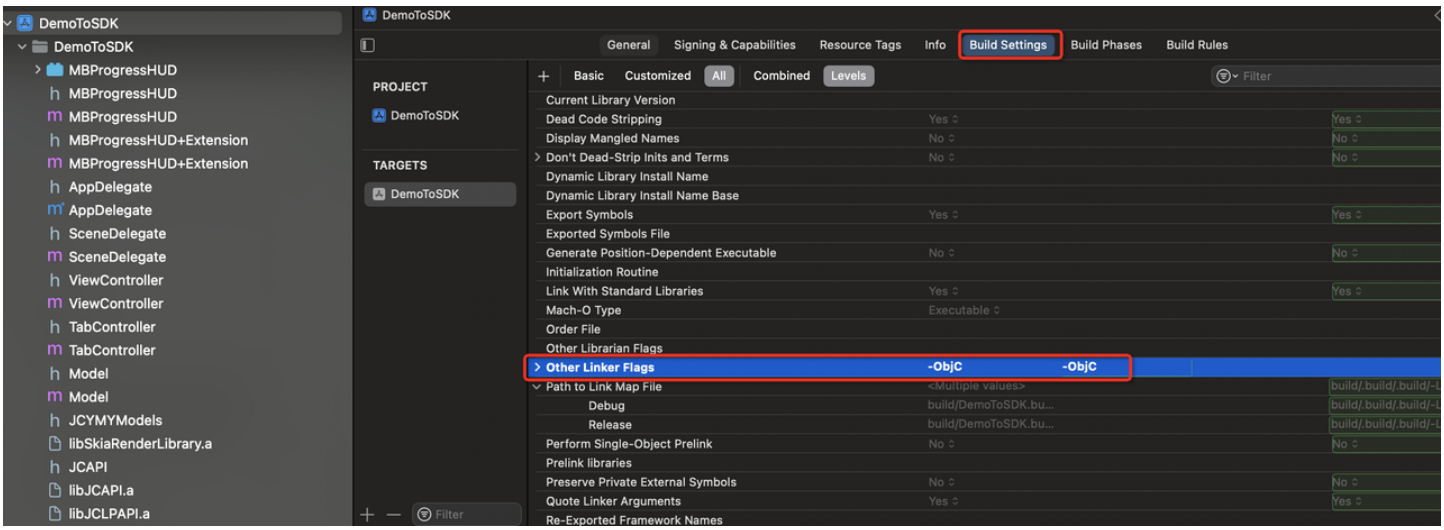
1.1 引入依赖项

注意:

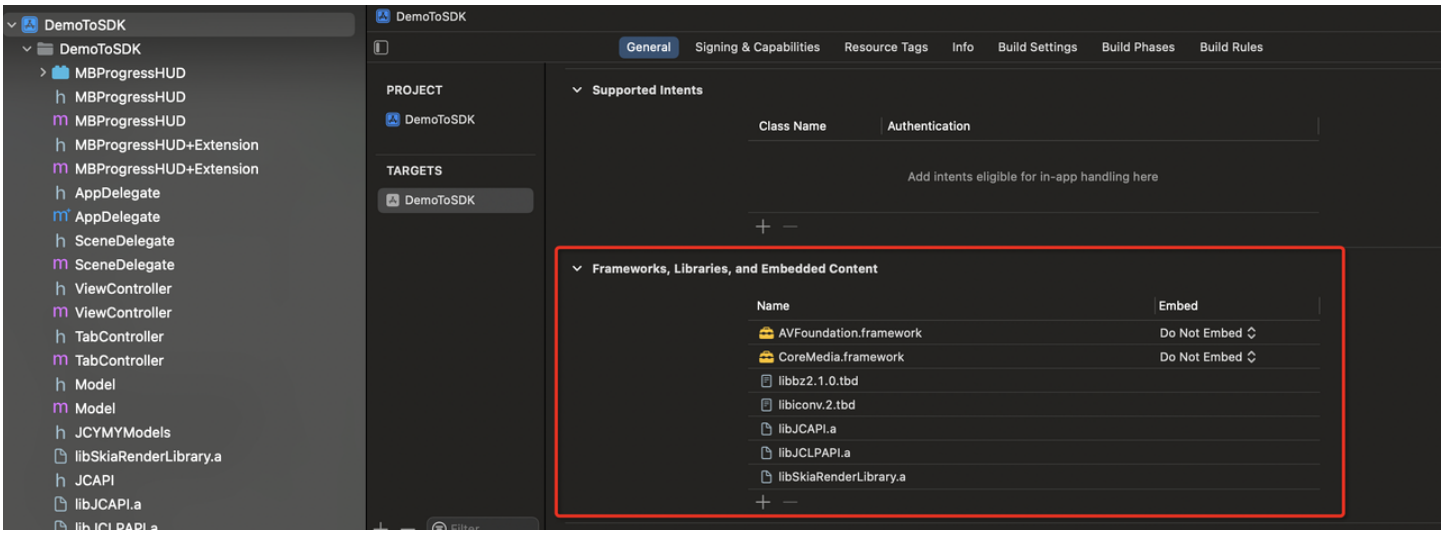
- 如接入的打印机型号不含B50/B11系列打印机，无需引入JCLPAPI.a
- 如无需自定义字体，则无需引入font文件夹

1.2 工程配置

1.2.1 oc反射配置



1.2.2 依赖配置



1.2.3 权限声明

需要在工程中info.plist文件中声明蓝牙权限，如图：

二、接口调用流程示例代码

2.1 搜索打印机

2.1.1 搜索蓝牙打印机

代码块

```
// 调用蓝牙打印机扫描方法
[JCAPI scanBluetoothPrinter:^(NSArray *scanedPrinterNames) {
    // 移除已有数据
    [weakSelf.datas removeAllObjects];

    // 遍历扫描到的蓝牙打印机名字
    for (NSString *name in scanedPrinterNames) {
        // 检查名字长度，如果不在指定范围内，跳过
        if (name.length < 6 || name.length > 20) {
            continue;
        }

        // 创建一个 Model 对象并设置其属性
        Model *m = [[Model alloc] init];
        m.name = name;

        // 将 Model 对象添加到数据数组
        [weakSelf.datas addObject:m];
    }
```

```
20  }];
```

2.1.2 搜索WIFI打印机

代码块

```
1   // 调用Wi-Fi打印机扫描方法
2   [JCAPI scanWifiPrinter:^(NSArray *scanedPrinterNames1) {
3       // 隐藏进度指示器（hud）
4       [weakSelf.hud hideAnimated:NO];
5       UIStoryboard *main = [UIStoryboard storyboardWithName:@"Main" bundle:nil];
6       // 实例化名为"TabController"的视图控制器
7       TabController *vc = [main
    instantiateViewControllerWithIdentifier:@"TabController"];
8       // 将数据（weakSelf.datas）传递给视图控制器
9       vc.datas = weakSelf.datas;
10      // 将该视图控制器添加为当前视图控制器的子视图控制器
11      [weakSelf addChildViewController:vc];
12      // 将视图控制器的视图添加到当前视图中
13      [weakSelf.view addSubview:vc.view];
14  }];
```

2.2 连接打印机

2.2.1 连接蓝牙打印机

代码块

```
1   //蓝牙连接指定名称的打印机
2   [JCAPI openPrinter:connectName completion:^(BOOL isSuccess) {
3       NSLOG(@"连接-%@",isSuccess?@"成功":@"失败");
4   }];
```

2.2.2 连接WIFI打印机

代码块

```
1   // WIFI连接指定IP的打印机（替换为您的打印机的实际IP 地址）
2   NSString *printerIP = @"192.168.1.100";
3   [JCAPI openPrinterHost:printerIP completion:^(BOOL isSuccess) {
4       NSLOG(@"连接-%@",isSuccess?@"成功":@"失败");
5   }];
```

2.3 调用打印相关方法

代码块

```
//字体名称
NSString *fontName = @"SourceHanSans-Regular";
//字体后缀
NSString *fontExtension = @"ttc";
//字体路径
NSString *fontPath = [[NSBundle mainBundle] pathForResource:fontName ofType:fontExtension];
//初始化图像库
[JCAPI initImageProcessing:fontPath error:nil];
//获取打印过程中的错误信息
[JCAPI getPrintingErrorInfo:^(NSString *printInfo) {
    if([printInfo isEqualToString:@"19"]){
        //纸张错误，可继续打印不处理
    }
    else{
        weakSelf.msgView.hidden = YES;
        MBProgressHUD *hub = [MBProgressHUD showMessage:printInfo];
        [hub hideAnimated:YES afterDelay:2.f];
    }
}];
//总打印份数
int count = 1;
//打印进度获取
[JCAPI getPrintingCountInfo:^(NSDictionary *printDicInfo) {
    NSString *totalCount = [printDicInfo valueForKey:@"totalCount"];
    if(totalCount.intValue == count){
        [JCAPI endPrint:^(BOOL isSuccess) {
            if(isSuccess)
                weakSelf.msgView.hidden = YES;
        }];
    }
}];
//设置SDK缓存
[JCAPI setPrintWithCache:YES];
//设置总打印份数
[JCAPI setTotalQuantityOfPrints:count];
//浓度
int blackRules = 3;
//纸张类型
int paperStyle = 1;
//开启打印任务
[JCAPI startJob:blackRules withPaperStyle:paperStyle withCompletion:^(BOOL isSuccess) {
```

```
    if(isSuccess){
            //设置画布尺寸
            [JCAPI initDrawingBoard:50 withHeight:30 withHorizontalShift:0 withVe                        rticalShift:0 rotate:0 font:@""];
            //绘制文本
            [JCAPI drawLableText:7.5 withY:5.0 withWidth:40.5 withHeight:6.5 with                        String:@"F金银花开植物饮料" withFontFamily:@"" withFontSize:3.5 withRot                        ate:0 withTextAlignHorizonral:0 withTextAlignVertical:1 withLineMode:
                    1 withLetterSpacing:0 withLineSpacing:1 withFontStyle:@[@0,@0,@0,@
                                0]];
            //
            //            [JCAPI DrawLableGraph:2 withY:2 withWidth:46 withHeight:26 withLineWidth:0.5 withCornerRadius:0 withRotate:0 withGraphType:1 withLineType:1 withDashWidth: nil];

            //            [JCAPI drawLableQrCode:5 withY:16 withWidth:8 withHeight:8 withString:@"https://www.baidu.com" withRotate:0 withCodeType:31];
            //            [JCAPI drawLableQrCode:2 withY:9 withWidth:8 withHeight:8 withString:@"https://www.baidu.com" withRotate:0 withCodeType:31];



            //            [JCAPI drawLableBarCode:x withY:y withWidth:width withHeight:height withString:text withFontSize:fontSize withRotate:ro withCodeType:codeType withTextHeight:textHeight withTextPosition:textPosition];
            //            [JCAPI DrawLableLine:x withY:y withWidth:width withHeight:height withRotate:ro withLineType:lineType withDashWidth:[dic valueForKey:@"dashwidth"]];
            //            [JCAPI DrawLableGraph:x withY:y withWidth:width withHeight:height withLineWidth:lineWidth withCornerRadius:cornerRadius withRotate:ro withGraphType:graphType withLineType:lineType withDashWidth:[dic valueForKey:@"dashwidth"]];
                        //imageData 为图像base64数据
            //            [JCAPI DrawLableImage:0 withY:0 withWidth:50 withHeight:20 withImageData:imageData withRotate:0 withImageProcessingType:1 withImageProcessingValue:127];
        [JCAPI commit:[JCAPI GenerateLableJson] withOnePageNumbers:1 withComp                    lete:^(BOOL isSuccess) {
            NSLog(@"----第一页数据");
            //                if(isSuccess){

            //                }
        }];
        }
        }];

```

```
69
70
71
                        }];
```

三、链接打印机相关方法

3.1 搜索蓝牙打印机

代码块

```
1  /**
2   扫描附近的蓝牙打印机。
3
4   该方法用于扫描附近的蓝牙打印机，并通过回调返回扫描到的打印机名称列表。
5
6   @param completion 扫描完成回调块。扫描完成后，将调用此回调并传递扫描到的蓝牙打印机名称数组。
7           数组 `scanedPrinterNames` 包含了扫描到的蓝牙打印机名称。如果未扫描到任何打印机，数组为空。
8   */
9  + (void)scanBluetoothPrinter:(void(^)(NSArray *scanedPrinterNames))completion;
```

3.2 连接蓝牙打印机

代码块

```
1  /**
2   蓝牙连接指定名称的打印机。
3
4   该方法用于与指定名称的蓝牙打印机进行连接。连接状态的变化会通过传递的回调进行通知。
5
6   @param printerName 要连接的蓝牙打印机的名称。
7   @param completion 连接状态回调块。当连接状态发生变化时，将调用此回调并传递连接状态的结果。
8           参数 `isSuccess` 代表是否成功连接打印机，YES 表示连接成功，NO 表示连接失败。
9   */
10 + (void)openPrinter:(NSString *)printerName
11          completion:(DidOpened_Printer_Block)completion;
```

3.3 搜索WIFI打印机

代码块

```
/**
 扫描附近的 Wi-Fi 打印机。

 该方法用于扫描附近的 Wi-Fi 打印机，并通过回调返回扫描到的打印机信息列表。

 @param completion 扫描完成回调块。扫描完成后，将调用此回调并传递扫描到的 Wi-Fi 打印机信息数组。
        数组 `scanedPrinterNames` 包含了扫描到的 Wi-Fi 打印机信息。每个元素是一个字典，包含以下字段：
        - `ipAdd`：打印机的 IP 地址。
        - `bleName`：蓝牙名字。
        - `port`：连接端口。
        - `availableClient`：可用客户端连接数。
 */
+ (void)scanWifiPrinter:(void(^)(NSArray *scanedPrinterNames))completion;
```

3.4 搜索WIFI打印机（可设置超时时间）

代码块

```
/**
 扫描附近的 Wi-Fi 打印机。

 该方法用于在指定的超时时间内扫描附近的 Wi-Fi 打印机，并通过回调返回扫描到的打印机名称列表。

 @param timeout 扫描超时时间，单位为秒。在这段时间内进行扫描操作。
 @param completion 扫描完成回调块。扫描完成后，将调用此回调并传递扫描到的 Wi-Fi 打印机名称数组。
        数组 `scanedPrinterNames` 包含了扫描到的 Wi-Fi 打印机名称。如果未扫描到任何打印机，数组为空。
 */
+ (void)scanWifiPrinter:(float)timeout withCompletion:(void(^)(NSArray *scanedPrinterNames))completion;
```

3.5 获取手机当前连接的WIFI名称

代码块

```
/**
```

```
 2    获取手机当前连接的Wi-Fi名称。
 3
 4    该方法用于获取手机当前连接的Wi-Fi的名称。
 5
 6    @return 返回手机当前连接的Wi-Fi名称。
 7    */
 8   + (NSString *)connectingWifiName;
```

3.6 配置打印机连接手机当前连接WIFI

代码块

```
 1   /**
 2   配置打印机连接手机当前连接wifi。
 3
 4    @param    wifiName          wifi账号（非必须）
 5    @param    password          wifi密码。
 6    @param    completion        配置打印机连接Wi-Fi是否成功。
 7    */
 8   + (void)configurationWifi:(NSString *)wifiName
 9                    password:(NSString *)password
10                  completion:(PRINT_DIC_INFO)completion;
```

3.7 获取打印机Wi-Fi配网信息

代码块

```
 1   /**
 2    获取打印机Wi-Fi配网信息。
 3
 4    该方法用于获取Wi-Fi配网信息，通常返回Wi-Fi名称。
 5
 6    @param completion Wi-Fi名称的回调。
 7    */
 8   + (void)getWifiConfiguration:(PRINT_DIC_INFO)completion;
```

3.8 连接WIFI打印机

代码块

```
 1    /**
 2    连接指定IP的打印机并进行 Wi-Fi 连接。
 3    该方法用于与指定 IP 地址的打印机建立 Wi-Fi 连接。连接状态的变化会通过传递的回调进行通
     知。
 4
```

```
 *  @param host 打印机的 IP 地址，用于指定要连接的打印机。
 *  @param completion 连接状态回调块。当连接状态发生变化时，将调用此回调并传递连接状态的结果。
 *          参数 `isSuccess` 代表是否成功连接打印机，YES 表示连接成功，NO 表示连接失败。
 */
+ (void)openPrinterHost:(NSString *)host
             completion:(DidOpened_Printer_Block)completion;
```

3.9 连接WIFI打印机（指定端口）

代码块

```
/**
 连接指定IP的打印机并进行 Wi-Fi 连接。
 该方法用于与指定 IP 地址的打印机建立 Wi-Fi 连接。连接状态的变化会通过传递的回调进行通知。

 @param host 打印机的 IP 地址，用于指定要连接的打印机。
 @param completion 连接状态回调块。当连接状态发生变化时，将调用此回调并传递连接状态的结果。
         参数 `isSuccess` 代表是否成功连接打印机，YES 表示连接成功，NO 表示连接失败。
 */
+ (void)openPrinterHost:(NSString *)host
             completion:(DidOpened_Printer_Block)completion;
```

3.10 关闭打印机

代码块

```
/**
 关闭当前打开的打印机连接。

 该方法用于关闭当前已经打开的打印机连接。在执行此操作后，将触发 `openPrinter:completion:` 方法的 `completion(NO)` 回调。

 注意：调用此方法会中断与打印机的连接。

 */
+ (void)closePrinter;
```

3.11 获取当前连接打印机名称

代码块

```
/**
 获取当前连接的打印机名称（蓝牙或 Wi-Fi）。

 该方法用于获取当前已连接的打印机的名称。对于 Wi-Fi 连接，返回的是打印机的 IP 地址。

 @return 当前连接的打印机名称。如果没有连接打印机，则返回 nil。
 */
+ (NSString *)connectingPrinterName;
```

3.12 获取当前连接状态

代码块

```
/**
 获取当前的蓝牙/Wi-Fi连接状态。

 该方法用于获取当前设备的蓝牙和 Wi-Fi 连接状态。

 @return 返回值为整型，表示连接状态。0 表示无连接，1 表示连接蓝牙，2 表示连接 Wi-Fi。
 */
+ (int)isConnectingState;
```

四、打印机参数设置与获取相关方法

4.1 打印机异常上报

代码块

```
/**
 蓝牙/Wi-Fi异常接收(连接成功后调用)。

 @param   error           打印异常: 1:盒盖打开,
                                    2:缺纸,
                                    3:电量不足,
                                    4:电池异常,
                                    5:手动停止,
                                    6:数据错误,
                                        (提交打印数据失败-B3/图像生成失败/发送数据错误,
打印机校验不通过打印机返回)
                                    7:温度过高,
                                    8:出纸异常,
 9-打印忙碌(当前正在转动马达(正在打印中或者走纸)/打印机正在升级固件)
```

```
    10-没有检测到打印头
    11-环境温度过低
    12.打印头未锁紧
    13-未检测到碳带
    14-不匹配的碳带
    15-用完的碳带
    16-不支持的纸张类型
    17-设置纸张失败
    18-设置打印模式失败
    19-设置打印浓度失败（允许打印,仅上报异常）
    20-写入Rfid失败
    21-边距设置错误
    (边距必须大于0，上边距+下边距必须小于画板高度，左边距+右边距必须小于画板宽度)
    22-通讯异常（超时，打印机指令一直拒绝）
    23-打印机断开
    24-画板参数设置错误
    25-旋转角度参数错误
    26-json参数错误(pc)
    27-出纸异常（关闭上盖检测）
    28-检查纸张类型
    29-RFID标签进行非RFID模式打印时
    30-浓度设置不支持
    31-不支持的打印模式
    32-标签材质设置失败(材质设置超时或者失败，不阻断正常打印)
    33-不支持的标签材质设置(阻断正常打印)
    34-打印机异常(阻断正常打印)
    35-切刀异常(T2阻断正常打印)
    36-缺纸(T2未放纸)
    37-打印机异常(T2无法通过指令恢复，需要手动按打印机)
    50-非法标签
    51-非法碳带和标签
    */
+ (void)getPrintingErrorInfo:(PRINT_INFO)error;
```

4.2 打印机打印页数获取

代码块

```
/**
 蓝牙/Wi-Fi打价器打印完成的份数(只对打价器有效，可能部分丢失，app做超时重置状态)。

 @param   count            打印完成的份数（在发生异常后不会返回）
 @{
    @"totalCount":@"总打印的张数计数" //返回必带的key
    @"pageCount":@"当前打印第PageNo页的第几份" //非必带
    @"pageNO":@"当前打印第几页"。 //非必带
```

```
    @"tid":@"写入rfid返回的tid码"  //非必带
    @"carbonUsed":@"碳带使用量，单位毫米"  //非必带
 }
 */
+ (void)getPrintingCountInfo:(PRINT_DIC_INFO)count;
```

4.3 监听打印机状态变化

代码块

```
/**
监听打印机状态变化

 @param   completion
 @{
    @"1": 盒盖状态-0打开/1关闭
    @"2": 电量等级变化-1/2/3/4
    @"3": 是否装有纸张-0没有/1有
    @"5": 碳带状态-0无碳带/1有碳带
    @“6”: wifi信号强度
 }
 @return  是否支持监听打印机状态变化：YES:支持、NO:不支持
 */
+ (BOOL)getPrintStatusChange:(PRINT_DIC_INFO)completion;
```

4.4 获取打印机内安装的标签尺寸

代码块

```
/**
 获取打印机内安装的标签尺寸（目前仅支持M2机型，固件版本V1.24以上版本）
 注意事项:statusCode为0，paperType参数不为0时读取的参数有效
 @{@"statusCode":@"0",
    @"result":@{@"gapHeightPixel":arrs[0],//间隙高度(黑标高度)(单位像素)
            @"totalHeightPixel":arrs[1],//纸张高度(包含间隙)(单位像素)
            @"paperType":arrs[2],//纸张类型 ： 1:间隙纸； 2:黑标纸； 3:连续纸； 4:定孔纸； 5:透明纸； 6:标牌;10:黑标间隙纸;
            @"gapHeight":arrs[3],//间隙高度(黑标高度)(单位毫米)
            @"totalHeight":arrs[4],//纸张高度(包含间隙)(单位毫米)
            @"paperWidthPixel":arrs[5],//纸张宽度(包含间隙)(单位像素)
            @"paperWidth":arrs[6],//纸张宽度(包含间隙)(单位毫米)
            @"direction":arrs[7], //尾巴方向1上2下3左4右（暂不支持）
            @"tailLengthPixel":arrs[8],//尾巴长度(单位像素)
            @"tailLength":arrs[9]}} //尾巴长度(单位毫米)
```

```
 */
+ (void)getPaperInfo:(PRINT_DIC_INFO)completion;
```

五、条码类型与回调信息说明

5.1 条码类型

代码块

```
| CODEBAR                                | JCBarcodeFormatCodebar      |
| Code39                                 | JCBarcodeFormatCode39       |
| Code93                                 | JCBarcodeFormatCode93       |
| Code128                                | JCBarcodeFormatCode128      |
| EAN-8                                  | JCBarcodeFormatEan8         |
| EAN-13                                 | JCBarcodeFormatEan13        |
| ITF(Interleaved Two of Five)           | JCBarcodeFormatITF          |
| UPC-A                                  | JCBarcodeFormatUPCA         |
| UPC-E                                  | JCBarcodeFormatUPCE         |
```

六、调用绘制/打印接口

6.1 初始化图像库

代码块

```
/**
 初始化图像库。

 该方法用于设置字体文件夹的路径，以供后续的图像处理操作。

 @param fontFamilyPath 字体文件夹的完整路径。
 @param error 用于接收错误信息的 NSError 对象的指针。如果设置字体路径时发生错误，将返回相应的错误信息。
 */
+(void) initImageProcessing:(NSString *) fontFamilyPath error:(NSError **)error;
```

6.2.1 开始绘制（已废弃）

代码块

```
/**
 初始化绘制画板。

 该方法用于初始化一个绘制画板，指定宽度、高度、水平偏移、竖直偏移、旋转角度以及可选的字体路径。

 @param width 画板的宽度（毫米）。
 @param height 画板的高度（毫米）。
 @param horizontalShift 画板的水平偏移（毫米）（暂不生效）。
 @param verticalShift 画板的竖直偏移（毫米）（暂不生效）。
 @param rotate 画板的旋转角度，通常为 0。
 @param font 字体路径，可选参数，如果不需要指定字体，可以输入 nil。（暂不生效）
 */
+(void)initDrawingBoard:(float)width
             withHeight:(float)height
   withHorizontalShift:(float)horizontalShift
     withVerticalShift:(float)verticalShift
                rotate:(int) rotate
                  font:(NSString*)font;
```

6.2.2 开始绘制

代码块

```
/**
 初始化绘制画板。

 该方法用于初始化一个绘制画板，指定宽度、高度、水平偏移、竖直偏移、旋转角度以及可选的字体路径。

 @param width 画板的宽度（毫米）。
 @param height 画板的高度（毫米）。
 @param horizontalShift 画板的水平偏移（毫米）（暂不生效）。
 @param verticalShift 画板的竖直偏移（毫米）（暂不生效）。
 @param rotate 画板的旋转角度，通常为 0。
 @param font 字体路径，可选参数，如果不需要指定字体，可以输入 nil。
 */
+(void)initDrawingBoard:(float)width
             withHeight:(float)height
   withHorizontalShift:(float)horizontalShift
     withVerticalShift:(float)verticalShift
                rotate:(int) rotate
                 fontArray:(NSArray<NSString*> *)fonts;
```

6.3 绘制文本

代码块

```
/**
 绘制文本。

 该方法用于在绘制画板上绘制文本，可以指定文本的位置、尺寸、内容、字体、字体大小、旋转角度、对齐方式、换行方式以及字体样式。

 @param x 水平起点（毫米）。
 @param y 竖直起点（毫米）。
 @param w 宽度（毫米）。
 @param h 高度（毫米）。
 @param text 文本内容。
 @param fontFamily 字体名称。
 @param fontSize 字体大小。
 @param rotate 旋转角度。
 @param textAlignHorizonral 文本水平对齐方式：0（左对齐）、1（居中对齐）、2（右对齐）。
 @param textAlignVertical 文本竖直对齐方式：0（顶对齐）、1（垂直居中）、2（底对齐）。
 @param lineMode 换行方式。1-宽高固定，内容大小自适应，2-宽度固定，高度自适应，3-宽高固定，超出内容用省略号表示，4-宽高固定，超出内容直接裁切，6-宽高固定，内容超过预设的宽高时自动缩小
 @param letterSpacing 字体间隔，单位：毫米
 @param lineSpacing 行间隔，单位：毫米
 @param fontStyles 字体样式，为包含布尔值的数组，通常包括加粗、斜体、下划线、删除下划线。

 @return 返回布尔值，表示文本是否成功绘制。

 @note
 文本绘制之前，请确保先调用此方法以初始化图像库。
 */
+(BOOL)drawLableText:(float)x
               withY:(float)y
           withWidth:(float)w
          withHeight:(float)h
          withString:(NSString *)text
      withFontFamily:(NSString *)fontFamily
        withFontSize:(float)fontSize
          withRotate:(int)rotate
withTextAlignHorizonral:(int)textAlignHorizonral
withTextAlignVertical:(int)textAlignVertical
        withLineMode:(int)lineMode
   withLetterSpacing:(float)letterSpacing
     withLineSpacing:(float)lineSpacing
```

```
39          withFontStyle:(NSArray <NSNumber *>*)fontStyles;
```

6.4 绘制一维码

代码块

```
1   /**
2    绘制一维码。
3
4    该方法用于在绘制画板上绘制一维码（条码），可以指定条码的位置、尺寸、内容、字号、旋转角
    度、类型以及相关文本信息。
5
6    @param x 水平坐标（毫米）。
7    @param y 垂直坐标（毫米）。
8    @param w 条码宽度（毫米）。
9    @param h 条码高度（毫米）（含文本高度）。
10   @param text 条码内容。
11   @param fontSize 文本字号。
12   @param rotate 旋转角度，仅支持 0, 90, 180, 270。
13   @param codeType 一维码类型:
14     - 20: CODE128
15     - 21: UPC-A
16     - 22: UPC-E
17     - 23: EAN8
18     - 24: EAN13
19     - 25: CODE93
20     - 26: CODE39
21     - 27: CODEBAR
22     - 28: ITF25
23   @param textHeight 文本高度（毫米）。
24   @param textPosition 一维码文字识别码显示位置:
25     - 0: 下方显示
26     - 1: 上方显示
27     - 2: 不显示
28
29   @return 返回布尔值，表示条码是否成功绘制。
30
31   @note
32   一维码绘制之前，请确保先调用此方法以初始化图像库。
33   */
34  +(BOOL)drawLableBarCode:(float)x
35                     withY:(float)y
36                 withWidth:(float)w
37                withHeight:(float)h
38                withString:(NSString *)text
39              withFontSize:(float)fontSize
```

```
                withRotate:(int)rotate
              withCodeType:(int)codeType
            withTextHeight:(float)textHeight
          withTextPosition:(int)textPosition;
```

6.5 绘制二维码

代码块

```
/**
 绘制二维码。

 该方法用于在绘制画板上绘制二维码，可以指定二维码的位置、尺寸、内容、旋转角度、类型。

 @param x 水平坐标（毫米）。
 @param y 垂直坐标（毫米）。
 @param w 二维码宽度（毫米）。
 @param h 二维码高度（毫米）。
 @param text 二维码内容。
 @param rotate 旋转角度，仅支持 0, 90, 180, 270。
 @param codeType 二维码类型：
   - 31: QR_CODE
   - 32: PDF417
   - 33: DATA_MATRIX
   - 34: AZTEC

 @return 返回布尔值，表示二维码是否成功绘制。
 */
+(BOOL)drawLableQrCode:(float)x
                 withY:(float)y
             withWidth:(float)w
            withHeight:(float)h
            withString:(NSString *)text
            withRotate:(int)rotate
          withCodeType:(int)codeType;
```

6.6 绘制线条

代码块

```
/**
 绘制线条。

```

```
 该方法用于在绘制画板上绘制线条，可以指定线条的位置、尺寸、旋转角度、类型以及虚线的宽度和样式。

 @param x 水平坐标（毫米）。
 @param y 垂直坐标（毫米）。
 @param w 线条宽度（毫米）。
 @param h 线条高度（毫米）。
 @param rotate 旋转角度，仅支持 0, 90, 180, 270。
 @param lineType 线条类型：
   - 1: 实线
   - 2: 虚线类型，虚实比例 1:1。
 @param dashWidth 虚线的宽度，为包含两个数字的数组，表示实线段长度和空线段长度。

 @return 返回布尔值，表示线条是否成功绘制。
 */
+(BOOL)DrawLableLine:(float)x
               withY:(float)y
           withWidth:(float)w
          withHeight:(float)h
          withRotate:(int)rotate
       withLineType:(int)lineType
      withDashWidth:(NSArray <NSNumber *>*)dashWidth;
```

6.7 绘制形状

代码块

```
/**
 绘制形状。

 该方法用于在绘制画板上绘制形状，可以指定形状的位置、尺寸、线条宽度、圆角、旋转角度、类型以及线条的样式。

 @param x 水平坐标（毫米）。
 @param y 垂直坐标（毫米）。
 @param w 形状宽度（毫米）。
 @param h 形状高度（毫米）。
 @param lineWidth 线条宽度（毫米）。
 @param cornerRadius 图像圆角（毫米）。
 @param rotate 旋转角度，仅支持 0, 90, 180, 270。
 @param graphType 图形类型，可选值：1-圆，2-椭圆，3-矩形，4-圆角矩形
 @param lineType 线条类型：
   - 1: 实线
   - 2: 虚线类型，虚实比例 1:1。
 @param dashWidth 线条的宽度，为包含两个数字的数组，表示实线段长度和空线段长度。
```

```
18
19  @return 返回布尔值，表示形状是否成功绘制。
20  */
21 +(BOOL)DrawLableGraph:(float)x
22                 withY:(float)y
23             withWidth:(float)w
24            withHeight:(float)h
25         withLineWidth:(float)lineWidth
26      withCornerRadius:(float)cornerRadius
27            withRotate:(int)rotate
28         withGraphType:(int)graphType
29          withLineType:(int)lineType
30         withDashWidth:(NSArray <NSNumber *>*)dashWidth;
```

6.8 绘制图片

代码块

```
1  /**
2   绘制图片。
3
4   该方法用于在绘制画板上绘制图片，可以指定图片的位置、尺寸、图像数据、旋转角度、处理算法以及阈值。
5
6   @param x 水平坐标（毫米）。
7   @param y 垂直坐标（毫米）。
8   @param w 图像宽度（毫米）。
9   @param h 图像高度（毫米）。
10  @param imageData 图像的 Base64 数据。（去除数据头）
11  @param rotate 旋转角度，仅支持 0, 90, 180, 270。
12  @param imageProcessingType 图像处理算法（默认1）。
13  @param imageProcessingValue 阈值（默认127）。
14
15  @return 返回布尔值，表示图片是否成功绘制。
16  */
17 +(BOOL)DrawLableImage:(float)x
18                 withY:(float)y
19             withWidth:(float)w
20            withHeight:(float)h
21         withImageData:(NSString *)imageData
22            withRotate:(int)rotate
23 withImageProcessingType:(int)imageProcessingType
24 withImageProcessingValue:(float)imageProcessingValue;
```

6.9 生成标签数据的JSON字符串

代码块

```
/**
 生成标签数据的 JSON 字符串。

 该方法用于生成标签数据的 JSON 字符串，以便提交给打印机进行打印。

 @return 返回生成的标签数据的 JSON 字符串。
 */
+(NSString *)GenerateLableJson;
```

6.10 获取画板上绘制内容的预览图

代码块

```
/**
 获取标签预览图像。

 该方法用于生成标签的预览图像，可以指定显示倍率和错误码。

 @param displayScale 显示倍率。
 @param error 返回的错误码，如果成功，error 为 nil。

 @return 返回生成的标签预览图像。
 */
+(UIImage *)generateImagePreviewImage:(float)displayScale error:(NSError
**)error;
```

6.11 设置SDK打印缓存

代码块

```
/**
 影响缓存和暂停功能，缓存最多5个任务，用以提高打印的连续型，提高打印体验
  否启动SDK缓存：YES:启动、NO:不启动
 */
+ (void)setPrintWithCache:(BOOL)startCache;
```

6.12 设置总张数

代码块

```
/**
 打印机打印前传入总打印份数

```

```
4    @param totalQuantityOfPrints 设置总打印份数，表示所有页面的打印份数之和。例如，如果你有3页需要打印，第一页打印3份，第二页打印2份，第三页打印5份，那么count的值应为10（3+2+5）。
5    */
6   + (void)setTotalQuantityOfPrints:(NSInteger)totalQuantityOfPrints;
```

6.13 开始打印任务

代码块

```
1   /**
2    准备打印任务。
3
4    该方法用于准备打印任务，设置打印浓度和纸张类型，并在打印完成后通过回调通知结果。
5
6    @param blackRules 打印浓度，根据不同打印机型号取值范围不同：
7                        - D11、D101、D110、H10、B16、B18、N1：1~3，默认2
8                        - B3S、B203、B1、B203、B31、B4、K3、K3W、K2、M2、M3：1~5，默认3
9                        - B50、B11、B50W、B32、Z401：1~15，默认8
10   @param paperStyle 纸张类型，可选值：
11                       - 1: 间隙纸
12                       - 2: 黑标纸
13                       - 3: 连续纸
14                       - 4: 定孔纸
15                       - 5: 透明纸
16                       - 6: 标牌
17                       - 10: 黑标间隙纸
18   @param completion 打印完成回调块。当打印任务完成时，将调用此回调并传递打印结果。
19   */
20  + (void)startJob:(int)blackRules
21    withPaperStyle:(int)paperStyle
22    withCompletion:(DidPrinted_Block)completion;
```

6.14 提交打印任务数据

代码块

```
1   /**
2    提交打印任务数据。
3
4    该方法用于提交打印数据，指定打印份数和回调处理。
5
6    @param printData 打印数据，通常为标签的 JSON 字符串。
7    @param onePageNumbers 用于指定当前页的打印份数。例如，如果你需要打印3页，第一页打印3份，第二页打印2份，第三页打印5份，那么在3次提交数据时，onePageNumbers值分别应为3，2，5。
```

```
 @param completion 提交数据完成回调，用于处理提交任务是否成功的结果，成功后方可进行下一次数据提交

 */
+ (void)commit:(NSString *)printData
withOnePageNumbers:(int)onePageNumbers
  withComplete:(DidPrinted_Block)completion;
```

6.15 提交打印任务数据（支持写入RFID）

代码块

```
/**
 提交打印任务数据。

 该方法用于提交打印数据，指定打印份数、写入RFID数据和回调处理。

 @param printData 打印数据，通常为标签的 JSON 字符串。
 @param onePageNumbers 用于指定当前页的打印份数。例如，如果你需要打印3页，第一页打印3份，第二页打印2份，第三页打印5份，那么在3次提交数据时，onePageNumbers值分别应为3，2，5。
 @param epcCode 要写入的RFID数据。可以为nil，表示不写入RFID数据。（仅支持B32R机型）
 @param completion 提交数据完成回调，用于处理提交任务是否成功的结果，成功后方可进行下一次数据提交
 */
+ (void)commit:(NSString *)printData
withOnePageNumbers:(int)onePageNumbers
       withEpc:(nullable NSString *)epcCode
  withComplete:(DidPrinted_Block)completion;
```

6.16 结束打印任务

代码块

```
/**
 蓝牙/Wi-Fi打印完成(打印完成后调用)。

 @param   completion      打印结束回调（在发生异常后不会返回）
 */
+ (void)endPrint:(DidPrinted_Block)completion;
```

6.17 取消打印

代码块

```
/**
```

```
 蓝牙/Wi-Fi取消打印(打印未完成调用)。

 @param   completion      打印结束回调（在发生异常后不会返回）
 */
+ (void)cancelJob:(DidPrinted_Block)completion;
```