PC 端 Web SDK React 接口说明文档 V4.0.6

文档修改记录

序号	版本号	修改内容	修改者	修改日期
1	v3.1.2	• 文档建立	周功成	2021/4/7
2	v3.1.4	• 更新 3.1.4 内容	石坤	2021/11/15
3	v3.1.8	• 兼容新机型 • 修复旋转后裁切异常 bug	石坤	2022/8/8
4	v3.1.8	更新接口封装	张彬	2023/6/7
5	v3.2.1	• 支持 K3 及 K3W 机型 • 增加 WIFI 相关接口	张彬	2023/10/12
6	v3.2.2	• 支持 M2 机型 • 增加 M2 相关说明	张彬	2023/10/30
7	v3.2.5	• 支持 B3S_P 机型 • 支持 B21S 机型 • 支持 B31 机型 • 更新图像库	张彬	2024/9/12
8	v4.0.3	• 支持 M3、K2、B21Pro 系列机型 • 完善错误码 • 新增绘制带 logo 二维码接口	张彬	2025/4/29

		• 提高 Websocket 通讯速度 5.Demo 支持黑标间隙纸		
9	v4.0.6	• 新增 closePrinter 接口 • 修复 Wifi 搜索接口 BUG • 修复历史遗留 WIFI 连接 BUG	张彬	2025/9/13

DEMO 目录结构

代码块

```
pc-react/
├── README.md                 # 项目说明文档
├── eslint.config.js          # ESLint配置文件
├── index.html                # 入口HTML文件
├── package.json              # 项目依赖和脚本配置
├── package-lock.json         # 依赖版本锁定文件
├── public/                   # 静态资源目录
├── src/                      # 源代码目录
│   ├── App.jsx               # 应用主组件
│   ├── Home.jsx              # 主页面组件
│   ├── HomeLogic.ts          # 主页业务逻辑类
│   ├── Print.ts              # 打印功能核心类
│   ├── PrintElementFactory.ts # 打印元素工厂类
│   ├── Socket.ts             # Socket通信类
│   ├── assets/               # 资源文件目录
│   ├── home.css              # 主页样式文件
│   ├── index.css             # 全局样式文件
│   ├── main.jsx              # 应用入口文件
│   └── printData/            # 打印数据目录
│       ├── Barcode.ts        # 条码打印数据
│       ├── Batch.ts          # 批量打印数据
│       ├── Combination.ts    # 组合打印数据
│       ├── Graph.ts          # 图形打印数据
│       ├── Img.ts            # 图片打印数据
│       ├── Line.ts           # 线条打印数据
│       ├── QrCode.ts         # 二维码打印数据
│       └── Text.ts           # 文本打印数据
├── tsconfig.json             # TypeScript配置文件
└── vite.config.js            # Vite构建配置文件
```

产品目的

JCAPI 接口为调用者提供易用的方法完成标签绘图、打印操作。本接口中提供了标贴的绘制方法，包括：文字、一维码、二维码，图形、线条、图像绘制，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，打印。方便用户在二次开发中调用接口，缩短开发周期，加快开发

打印机支持

支持打印机型号
B1
B203
B21 /B21_Pro/B21S
B3S / B3S_P
B31
B4
B11
K2
K3/K3W
B50/B50W
B32/Z401/B32R
M2
M3

准备工作

- 安装精臣打印服务（jcPrinterSdk.exe）
  - 前置：关闭杀毒软件（如 360，易误报）
  - 关键：**必须默认路径安装（C 盘）**

  - 注意：勿禁用服务开机启动

- 安装对应机型驱动

机型系列	系统要求
B50/B11	Win7/10/11 均需装驱动
其他机型	Win10/11 无需装，仅 Win7 需装

- 设备连接（2 种方式，不支持蓝牙）
  - USB 连接
    - 系统：仅支持 Windows
    - 驱动：可能需装（参考第 2 步）
    - 特别：**不支持驱动打印**（已用驱动打印需下载专用驱动
  - WIFI 连接
    - 机型：**仅支持 K3W 机型**
    - 系统：仅支持 Windows
    - 驱动：无需安装

一、初始化及打印调用流程、打印流程

1.1 初始化流程

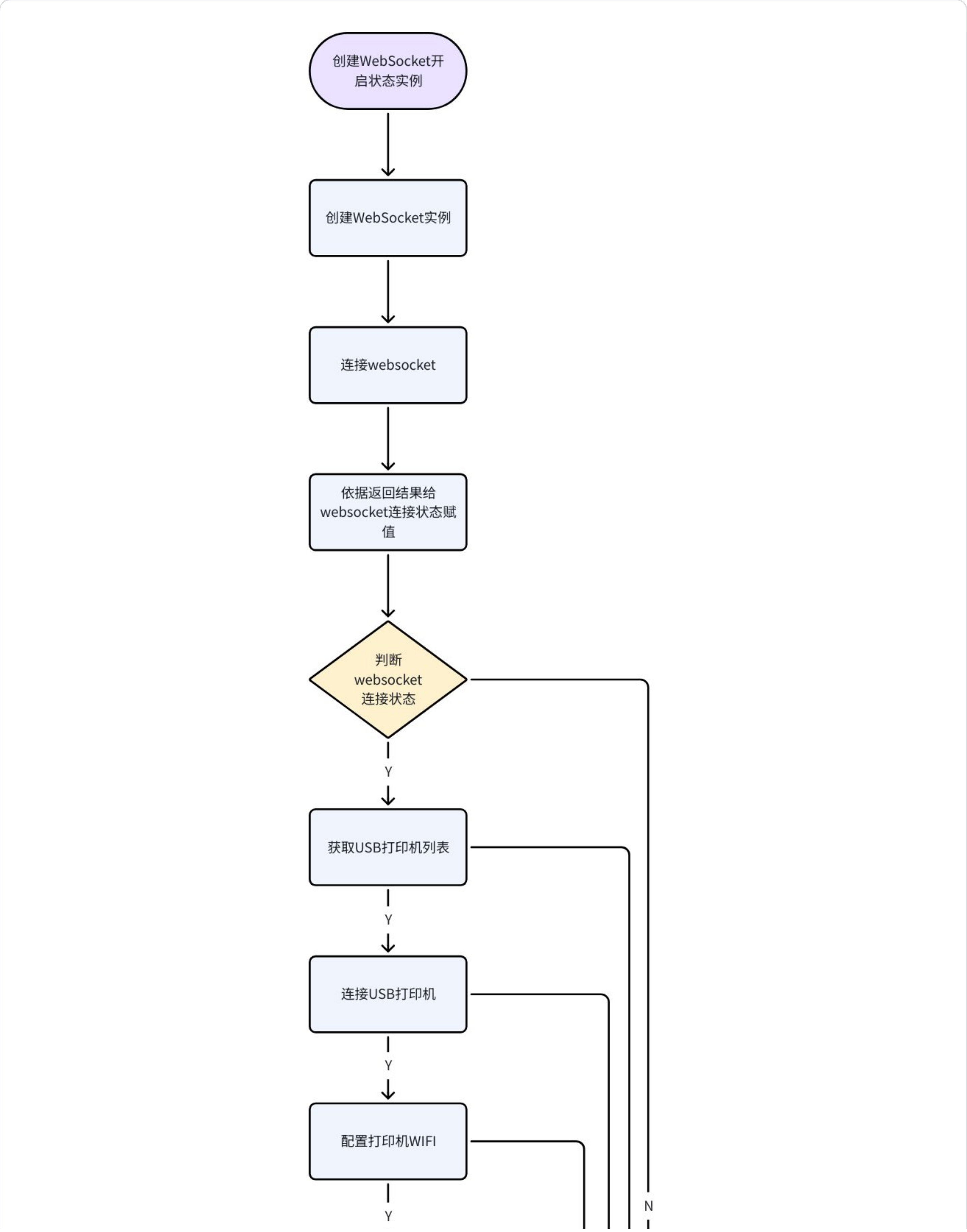
1.1.1 USB 打印初始化流程

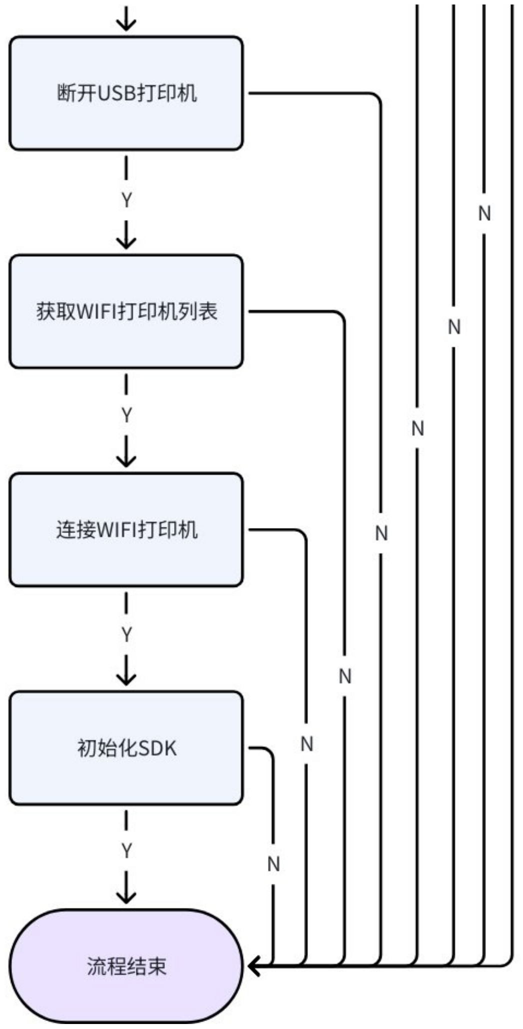
- WebSocket 建议页面加载时进行初始化，在 WebSocket 初始成功后回调中进行获取打印机、选择打印机、初始化 SDK 等操作
- 因为所有接口均为异步操作，调用下一接口需要验证当前接口结果后再执行下一接口
- 记录打印机列表获取状态、连接状态、初始化状态，打印机需要检查对应的状态

1.1.2 WIFI 打印初始化流程

- WebSocket 建议页面加载时进行初始化，在 WebSocket 初始成功后回调中进行获取打印机、选择打印机、初始化 SDK 等操作

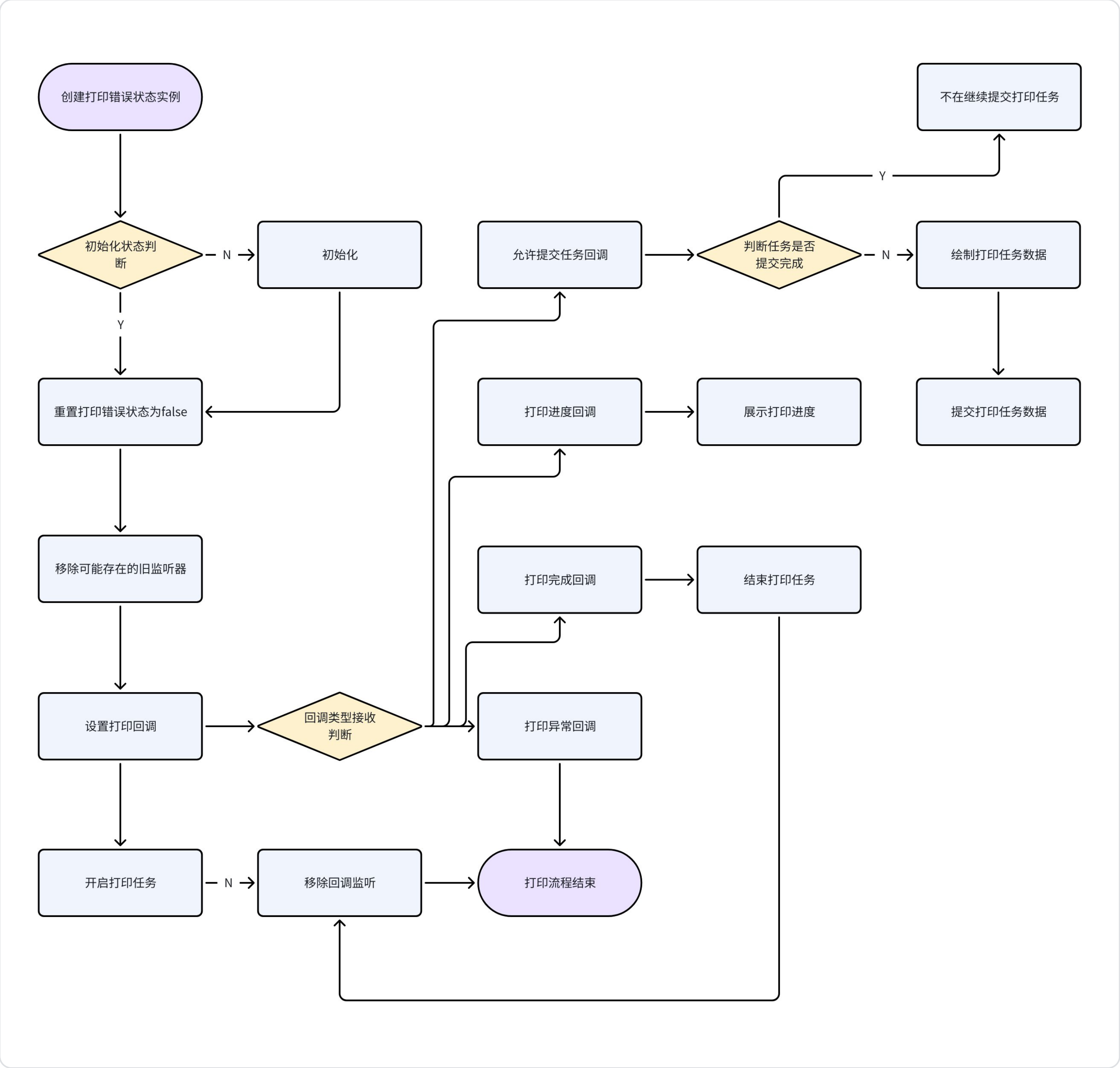
- 因为所有接口均为异步操作，调用下一接口需要验证当前接口结果后再执行下一接口
- 记录打印机列表获取状态、连接状态、初始化状态，打印机需要检查对应的状态
- **打印机 WIFI 配置成功后，后续直接搜索连接，无需多次进行配置（省略 USB 打印机获取、打印机连接、打印机网络配置）**





1.2 打印流程

- 打印前建议判断 WebSocket 是否初始化成功、SDK 是否初始化成功（包含初始化 SDK，获取打印机、选择打印机三个流程）
- 因为所有接口均为异步操作，除 WebSocket 初始化调用是在单独的回调接口中判断是否初始化成功外，其他接口应通过添加 await 关键字调用方法后，等待方法返回结果，解析返回的结果数据后进行判断再进行下一接口调用
- 打印回调监听会有多种回调，包含异常取消、页码回调，可参考流程图及 DEMO 进行处理



二、页面初始化相关接口

2.1 初始化打印服务及接口实例（包含打印机状态回调）

代码块

```
export default class Socket {
  /**
   * 打开 WebSocket 连接并返回一个解析为 WebSocket 实例的 Promise。
   *
   * @param {function} openChange - WebSocket 连接打开时要调用的回调函数。
   * @param {function} onMessageCallback - 接收到消息时要调用的回调函数。
   * @return {Promise} 一个解析为 WebSocket 实例的 Promise。
   */
  public open(openChange?: (isOpen: boolean) => void, onMessageCallback?: (msg: any) => void): Promise<{ e: Event, ws: Socket }> {}
}


export default class NMPrintSocket {
    constructor(printSocketData: Socket) {
        this.printSocket = printSocketData;
    }
}
```

代码块

```
export class HomeLogic {
    // WebSocket实例
    public socketData: Socket | null = null;

    // 打印Socket实例
    public nMPrintSocket: NMPrintSocket | null = null;
    public initialize(): void {
        // 创建Socket实例
        this.socketData = new Socket();

        // 打开WebSocket连接
        this.socketData.open(
            // 连接状态回调
            async (openBool) => {
                console.log("WebSocket连接状态:", openBool);
                // 更新套接字连接状态
                this.printSocketOpen = openBool;
                this._updateReactState();
            },
            // 消息处理回调
            (msg: any) => {
                // 处理设备状态回调
                if (msg.resultAck.callback !== undefined) {
                    const callbackName = msg.resultAck.callback.name;
                    const msgInfo = msg.resultAck.info;
                    if (callbackName === "onCoverStatusChange") {
                        console.log("盒盖状态", msgInfo.capStatus);
                    } else if (callbackName === "onElectricityChange") {
                        console.log("电池电量等级", msgInfo.power);
                    }
                }
            }
        );

        // 创建打印Socket实例
        this.nMPrintSocket = new NMPrintSocket(this.socketData);
        this._updateReactState();
    }
}
```

2.2 初始化 SDK initSdk

代码块

```
export default class NMPrintSocket {
   /**
    * 初始化SDK，在打印服务连接成功后调用此接口。
    * 在调用SDK的绘制接口之前，必须先调用此接口。
    *
    * @param {object} json - 包含必要参数的JSON对象,格式如下:
    *  {
    *   "fontDir": string, //字体文件目录，默认为""，暂不生效
    * }
    *
    * @return {Promise} 返回一个 Promise，解析为初始化SDK的结果
    */
    public initSdk(json: { fontDir: string }): Promise<any>
}
```

代码块

```
//初始化SDK参数JSON
{
  "fontDir": ""
}
//初始化成功返回JSON
{
  "apiName": "initSdk",
    "resultAck": {
      "errorCode": 0,
      "info": "initSdkApi ok!",
      "result": 0
    }
}

/**
```

```
 * 初始化SDK
 * 通过API调用初始化SDK
 */
public async init(): Promise<void> {
    if (!this.printSocketOpen || !this.nMPrintSocket) {
        return alert("打印服务未开启");
    }
    if (!this.onlineUsbBool && !this.onlineWifiBool) {
        return alert("打印机未连接");
    }
    try {
        const initRes = await this.nMPrintSocket.initSdk({ fontDir: "" });
        const result = JSON.parse(initRes.resultAck.errorCode);
        if (result === 0) {
            this.initBool = true;
            console.log("SDK初始化成功");
        } else {
            this.initBool = false;
            console.log("SDK初始化失败");
            alert("SDK初始化失败");
        }
    } catch (err) {
        console.error(err);
        this.initBool = false;
        alert("SDK初始化异常");
    }
    this._updateReactState();
}
```

2.3 获取 USB 打印机列表 getAllPrinters

代码块

```
export default class NMPrintSocket {
    /**
     * 获取所有当前PC上通过USB连接的精臣打印机
     *
     * @return {Promise} 返回一个Promise，解析为打印机列表。
     *
     * @description
     * 需要在打印服务连接成功后调用此函数。
     */
    public getAllPrinters(): Promise<any>
}
```

代码块

```
//返回结果
{
        "apiName": "getAllPrinters",
        "resultAck": {
                "errorCode": 0,
                "info": "{\"e623012991\":\"31\"}",//打印机名称及类型
                "result": "true"
        }
}

/**
 * 获取所有USB打印机
 * 通过API调用获取所有已连接的USB打印机
 */
public async getPrinters(): Promise<void> {
    if (!this.printSocketOpen || !this.nMPrintSocket) {
        // alert("打印服务未开启");
        console.log("打印服务未开启");
        return;
    }
    console.log("开始获取打印机");
    try {
        const allPrintersRes = await this.nMPrintSocket.getAllPrinters();
        console.log(allPrintersRes, "allPrintersRes");
        if (allPrintersRes.resultAck.errorCode === 0) {
            const allPrinters = JSON.parse(allPrintersRes.resultAck.info);
            this.usbPrinters = { ...allPrinters };
            this.usbSelectPrinter = Object.keys(this.usbPrinters)[0] || "";
            console.log("printers", this.usbPrinters);
        } else {
            this.usbPrinters = {};
            this.usbSelectPrinter = "";
            alert("没有在线的USB打印机");
        }
    } catch (err) {
        console.error(err);
        this.usbPrinters = {};
        this.usbSelectPrinter = "";
        alert("获取USB打印机列表失败");
    }
    this._updateReactState();
}
```

2.4 获取 WIFI 连接的打印机列表 scanWifiPrinter

代码块

```
export default class NMPrintSocket {
    /**
     * 搜索Wifi打印机
     *
     *
     * @return {Promise} 返回一个 Promise，解析为打印机Wifi配置信息
     *
     * @description
     * 需要在打印服务连接成功后调用此函数。
     */
    public scanWifiPrinter(): Promise<any>
}
```

代码块

```
//返回结果
{
        "apiName": "scanWifiPrinter",
        "resultAck": {
                "errorCode": 0,
                "info": "[{
      "deviceName":"K3W-E828013369",
                        "IP":"192.168.1.10",
                        "tcpPort":"9200",
                        "avaliableClient":"0"
          }]",
    "result": "true"
        }
}


/**
 * 扫描所有Wifi打印机
 * 通过API调用扫描所有可用的Wifi打印机
 */
public async scanWifiPrinters(): Promise<void> {
    if (!this.printSocketOpen || !this.nMPrintSocket) {
        alert("打印服务未开启");
        return;
    }
    try {
        const allPrintersRes = await this.nMPrintSocket.scanWifiPrinter();
        console.log("allPrintersRes", allPrintersRes);
        const errorCode = allPrintersRes.resultAck.errorCode;
        if (errorCode === 0) {
            const allPrinters: WifiPrinterInfo[] = allPrintersRes.resultAck.info;
            this.wifiPrinters = {};
            allPrinters.forEach((item) => {
                this.wifiPrinters[item.deviceName] = item.tcpPort.toString();
            });
            console.log("wifiPrinters", this.wifiPrinters);
            this.wifiSelectPrinter = Object.keys(this.wifiPrinters)[0] || "";
            console.log("wifiSelectPrinter", this.wifiSelectPrinter);
        } else {
            this.wifiPrinters = {};
            this.wifiSelectPrinter = "";
            alert("没有在线的Wifi打印机");
        }
    } catch (err) {
        console.error(err);
        this.wifiPrinters = {};
        this.wifiSelectPrinter = "";
        alert("扫描Wifi打印机列表失败");
    }
    this._updateReactState();
}
```

2.5 连接 USB 打印机 selectPrinter

代码块

```
export default class NMPrintSocket {
    /**
     * 选择并打开需要使用的打印机名称，及端口号
     *
     * @param {string} printerName - 打印机名称。
     * @param {number} port - 连接端口。
     * @return {Promise} 返回一个Promise，解析为连接结果
     *
     * @description
     * 需要在打印服务连接成功后调用此函数，建议在getAllPrinters调用成功后调用该接口，以保证传入的打印机名称和端口的打印机状态正常。。
     */
    public selectPrinter(printerName: string, port: number): Promise<any>
}
```

代码块

```
//返回数据示例
{
        "apiName": "selectPrinter",
        "resultAck": {
                "callback": {
                        "name": "onConnectSuccess",
                        "printerName": "e623012991"
                },
                "errorCode": 0,
                "info": "select printer ok!",
                "result": true
        }
}

/**
 * 连接选中的USB打印机
 * 通过API调用连接用户选择的USB打印机
 */
public async selectOnLineUsbPrinter(): Promise<void> {
    if (!this.printSocketOpen || !this.nMPrintSocket) {
        return alert("打印服务未开启");
    }
    if (!this.usbSelectPrinter) {
        alert("请先选择一个USB打印机");
        return;
    }
    try {
        console.log("this.usbSelectPrinter", this.usbSelectPrinter);
        console.log("this.usbPrinters[this.usbSelectPrinter]", this.usbPrinters[this.usbSelectPrinter]);
        const usbConnectRes = await this.nMPrintSocket.selectPrinter(this.usbSelectPrinter, parseInt(this.usbPrinters[this.usbSelectPrinter]));
        const result = JSON.parse(usbConnectRes.resultAck.errorCode);
        console.log("result", result);
        if (result === 0) {
            console.log("USB打印机连接成功");
            this.onlineUsbBool = true;
            this.onlineWifiBool = false;
        } else {
            console.log("USB打印机连接失败");
            this.onlineUsbBool = false;
            alert("USB打印机连接失败");
        }
        console.log("usbConnectRes", usbConnectRes);
    } catch (err) {
        console.error(err);
        this.onlineUsbBool = false;
        alert("连接USB打印机异常");
    }
    this._updateReactState();
}
```

2.6 连接 WIFI 打印机列表中的打印机 connectWifiPrinter

代码块

```
export default class NMPrintSocket {
    /**
     * 发送消息以选择打印机。
     *
     * @param {string} printerName - 打印机名称。
     * @param {number} tcpPort - 端口号。
     * @return {Promise} 返回连接结果
     *
     * @description
     * 需要在打印服务连接成功后调用此函数，建议在scanWifiPrinter调用成功的回调接口中调用该接口，保证传入的打印机名称和端口的打印机状态正常。
     * 注意：此函数仅能连接 WIFI 打印机列表中的打印机。
     */
    public connectWifiPrinter(printerName: string, tcpPort: number): Promise<any>
}
```

代码块

```
//示例返回成功数据
{
        "apiName": "selectPrinter",
        "resultAck": {
                "callback": {
                        "name": "onConnectSuccess",
                        "printerName": "e623012991"
                },
                "errorCode": 0,
                "info": "select printer ok!",
                "result": true
        }
}
//示例返回失败数据
{
  "apiName":"connectWifiPrinter",
  "resultAck":{
    "callback":{
      "name":"onDisConnect",
      "printerName":"K3_W-F612010061"
    },
    "errorCode":0,
    "info":"success",
    "result":false
  }
}

/**
 * 连接选中的Wifi打印机
 * 通过API调用连接用户选择的Wifi打印机
 */
public async selectOnLineWifiPrinter(): Promise<void> {
    if (!this.printSocketOpen || !this.nMPrintSocket) {
        return alert("打印服务未开启");
    }
    if (!this.wifiSelectPrinter || !this.wifiPrinters[this.wifiSelectPrinter]) {
        alert("请先选择一个有效的Wifi打印机");
        return;
    }
    try {
        const wifiConnectRes = await this.nMPrintSocket.connectWifiPrinter(
            this.wifiSelectPrinter,
            parseInt(this.wifiPrinters[this.wifiSelectPrinter])
        );
        const result = JSON.parse(wifiConnectRes.resultAck.errorCode);
        if (result) {
            console.log("Wifi打印机连接成功");
            this.onlineWifiBool = true;
            this.onlineUsbBool = false;
        } else {
            console.log("Wifi打印机连接失败");
            this.onlineWifiBool = false;
            alert("Wifi打印机连接失败");
        }
        console.log("wifiConnectRes", wifiConnectRes);
    } catch (err) {
        console.error(err);
        this.onlineWifiBool = false;
        alert("连接Wifi打印机异常");
    }
    this._updateReactState();
}
```

2.7 断开打印机连接closePrinter

代码块

```
export default class NMPrintSocket {
    /**
     * 断开打印机连接。
     *
     * @return {Promise} 返回一个Promise，解析为关闭结果
     */
    public closePrinter(): Promise<any>
}
```

2.8 配置打印机的 WIFI 信息 configurationWifi

代码块

```
export default class NMPrintSocket {
    /**
     * 配置打印机的Wifi网络
     *
     * @param {string} wifiName - wifi网络的名称。
     * @param {string} wifiPassword - wifi网络的密码。
     * @return {Promise} 返回一个 Promise，解析为打印机Wifi配置结果
     *
     * @description
     * 注意:仅支持2.4G频段网络，且需要在连接成功后配置。首次配置建议在USB连接成功后配置
     */
    public configurationWifi(wifiName: string, wifiPassword: string): Promise<any>
}
```

代码块

```
//示例返回数据
{
  "apiName":"configurationWifi",
  "resultAck":{
    "errorCode":0,
    "info":"configuration wifi printer ok!",
    "result":true
  }
}

```

```
/**
 * 配置打印机的Wifi网络
 * 通过API调用配置打印机的Wifi网络
 */
public async setWifiConfiguration(): Promise<void> {
    if (!this.printSocketOpen || !this.nMPrintSocket) {
        return alert("打印服务未开启");
    }
    try {
        if (this.wifiName.trim() !== "") {
            const wifiConfigurationResult = await this.nMPrintSocket.configurationWifi(
                this.wifiName.trim(),
                this.wifiPassword.trim()
            );
            console.log("wifiConfigurationResult", wifiConfigurationResult);
            const errorCode = wifiConfigurationResult.resultAck.errorCode;
            if (errorCode === 0) {
                alert("网络配置成功，请断开USB线缆后使用WIFI搜索连接打印机（PC需要和打印机在同一网络）");
            } else {
                alert("网络配置失败，错误码：" + errorCode);
            }
        } else {
            alert("wifi名称不得为空");
        }
    } catch (err) {
        console.error(err);
        alert("配置Wifi网络异常");
    }
}
```

2.9 获取打印机的 WIFI 相关配置 getWifiConfiguration

代码块

```
export default class NMPrintSocket {
    /**
     * 获取打印机的wifi配置。
     *
     * @return {Promise} 返回一个 Promise，解析为打印机Wifi配置信息
     */
    public getWifiConfiguration(): Promise<any>
}
```

代码块

```
//示例返回成功数据
 {
   "apiName":"getWifiConfiguration",
     "resultAck":{
     "errorCode":0,
     "info":"{
       \n\t\"wifiName\" : \"Test\"\n
       }\n",
      "result":"{
       \n\t\"wifiName\" : \"Test\"\n
       }\n"
   }
 }
//示例返回失败数据
 {
   "apiName":"getWifiConfiguration",
     "resultAck":{
     "errorCode":23,
     "info":"select printer connect first!",
     "result":false
   }
 }

/**
 * 获取当前打印机的Wifi配置信息
 * 通过API调用获取打印机的Wifi配置信息
 */
public async getWifiConfigurationInfo(): Promise<void> {
    if (!this.printSocketOpen || !this.nMPrintSocket) {
        return alert("打印服务未开启");
    }
    try {
        const wifiInfo = await this.nMPrintSocket.getWifiConfiguration();
        const errorCode = wifiInfo.resultAck.errorCode; // Assuming errorCode is a number directly

        if (errorCode === 0) {
            const info = JSON.parse(wifiInfo.resultAck.info);
            console.log("wifiInfo", info);
            alert("wifiInfo:" + JSON.stringify(info));
        } else {
            alert("wifiInfo:获取失败，错误码：" + errorCode);
        }
    } catch (err) {
        console.error(err);
        alert("获取Wifi配置信息异常");
    }
}
```

三、绘制打印数据相关接口

3.1 创建画板 InitDrawingBoard

代码块

```
export default class NMPrintSocket {
    /**
     * 初始化绘制画板
     *
     * @param {Object} json - 包含初始化绘制画板所需数据的JSON对象。格式如下:
     * {
     *   "width": number, // 画板的宽度，单位为mm
     *   "height": number, // 画板的高度，单位为mm
     *   "rotate": number, // 画板的旋转角度，仅支持0、90、180、270
     *   "path": string, // 字体文件的路径，默认为""，暂不生效
     *   "verticalShift": number, // 垂直偏移量，暂不生效
     *   "HorizontalShift": number // 水平偏移量，暂不生效
     * }
     * @return {Promise} 返回一个 Promise，解析为初始化绘制画板的结果
     *
     * @description
     * 增加接口说明:
     * 1.在调用绘制接口之前，必须先初始化SDK。
     * 2.绘制元素前，必须先初始化画板，否则会引起崩溃!
     * 3.初始化画板时会清空画板上次绘制的内容!
     */
    public InitDrawingBoard(json: String): Promise<any>
}
```

代码块

```
{
        "apiName": "InitDrawingBoard",
        "resultAck": {
                "errorCode": 0,
                "info": "init draw board success!",
                "result": 0
        }
}
```

```

/**
 * 初始化打印画布
 * 通过API调用初始化打印画布
 * @param params - 画布初始化参数
 * @returns 是否初始化成功
 */
private async initCanvas(params: String): Promise<boolean> {
    if (!this.printSocketOpen || !this.nMPrintSocket || !this.initBool) {
        return false;
    }
    console.log("初始化打印画布");
    try {
        const res = await this.nMPrintSocket.InitDrawingBoard(params);
        return res.resultAck.errorCode === 0;

    } catch (err) {
        console.error('画布初始化错误:', err);
        return false;
    }
}
```

3.2 绘制文本 DrawLabelText

代码块

```
export default class NMPrintSocket {
    /**
     * 绘制标签文本。
     * @param {object} json - 包含标签文本信息的JSON对象。
     *   JSON格式要求如下:
     *   - x: x轴坐标，单位mm
     *   - y: y轴坐标，单位mm
     *   - height: 文本高度，单位mm
     *   - width: 文本宽度，单位mm
     *   - value: 文本内容
     *   - fontFamily: 字体名称，暂不生效，使用默认字体思源黑体
     *   - rotate: 旋转角度，0:0, 1:90, 2:180, 3:270
     *   - fontSize: 字号，单位mm
     *   - textAlignHorizonral: 水平对齐方式: 0:左对齐 1:居中对齐 2:右对齐
     *   - textAlignVertical: 垂直对齐方式: 0:顶对齐 1:垂直居中 2:底对齐
     *   - letterSpacing: 字母之间的标准间隔，单位mm
     *   - lineSpacing: 行间距（倍距），默认1
     *   - lineMode: 1:宽高固定，内容大小自适应，预设宽高过大时字号放大，预设宽高过小时字号缩小，
     *     保证内容占据满预设宽高（字号/字符间距/行间距 按比例缩放）
     *     2:宽度固定，高度自适应
     *     4:宽高固定,超出内容直裁切
     *     6:宽高固定，内容超过预设的文本宽高自动缩放
     *     建议设置为6
     *   - fontStyle: 字体样式[加粗，斜体，下划线，删除下划线（预留）]
     *
     * @return {Promise} 返回一个 Promise，解析为绘制标签文本的结果
     * @description 绘制标签文本前必须先初始化画板
     */
    public DrawLabelText(json: String): Promise<any>
}
```

代码块

```
//返回数据示例
{
        "apiName": "DrawLableText",
        "resultAck": {
                "errorCode": 0,
                "info": "draw bar code success!",//此处返回信息有误，下个版本修复
                "result": 0
        }
}

/**
 * 打印元素处理方法
 * 支持多种打印元素类型，按顺序执行绘制操作
 * @param elements - 打印元素数组
 * @returns 是否全部元素处理成功
 */
private async processPrintElements(elements: any[]): Promise<boolean> {
    if (!this.nMPrintSocket) return false;
    console.log("elements", elements);
    for (const element of elements) {
        let res;
        switch (element.type) {
            case "text": // 文本打印
                res = await this.nMPrintSocket.DrawLabelText(element.json);
                break;
            case "qrCode": // 二维码打印
                res = await this.nMPrintSocket.DrawLabelQrCode(element.json);
                break;
            case "barCode": // 条形码打印
                res = await this.nMPrintSocket.DrawLabelBarCode(element.json);
                break;
            case "line": // 线条绘制
```

```
                res = await this.nMPrintSocket.DrawLabelLine(element.json);
                break;
            case "graph": // 图形绘制
                res = await this.nMPrintSocket.DrawLabelGraph(element.json);
                break;
            case "image": // 图像打印
                res = await this.nMPrintSocket.DrawLabelImage(element.json);
                break;
            default:
                console.error("Unsupported element type:", element.type);
                return false;
        }
        if (parseInt(JSON.parse(res.resultAck.errorCode)) !== 0) {
            console.error(`Failed to draw ${element.type}:`, res);
            return false;
        }
    }
    return true;
}
```

3.3 一维码绘制 DrawLabelBarCode

代码块

```
export default class NMPrintSocket {
    /**
     * 绘制一维码条形码。
     *
     * @param {Object} json - 包含一维码条形码信息的JSON对象。格式如下:
     * {
     *   "x": number, // x轴坐标，单位mm
     *   "y": number, // y轴坐标，单位mm
     *   "height": number, // 一维码宽度，单位mm
     *   "width": number, // 一维码高度，单位mm (包含文本高度)
     *   "value": string, // 一维码内容
     *   "codeType": number, // 条码类型:
     *                     // 20: CODE128
     *                     // 21: UPC-A
     *                     // 22: UPC-E
     *                     // 23: EAN8
     *                     // 24: EAN13
     *                     // 25: CODE93
     *                     // 26: CODE39
     *                     // 27: CODEBAR
     *                     // 28: ITF25
     *   "rotate": number, // 旋转角度，0: 0, 1: 90, 2: 180, 3: 270
     *   "fontSize": number, // 文本字号，单位mm，字号为0则文本不显示
     *   "textHeight": number, // 文本高度，单位mm，高度为0则文本不显示
     *   "textPosition": number // 一维码文字识别码显示位置：
     *                          // 0: 下方显示
     *                          // 1: 上方显示
     *                          // 2: 不显示
     * }
     *
     * @return {Promise} 返回一个 Promise，解析为绘制一维码条形码的结果
     *
     * @description
     * 1. 绘制元素前，必须先初始化画板
     */
    public DrawLabelBarCode(json: String): Promise<any>
}
```

代码块

```
//返回数据示例
{
        "apiName": "DrawLableBarCode",
        "resultAck": {
                "errorCode": 0,
                "info": "draw bar code success!",
                "result": 0
        }
}

/**
 * 打印元素处理方法
 * 支持多种打印元素类型，按顺序执行绘制操作
 * @param elements - 打印元素数组
 * @returns 是否全部元素处理成功
 */
private async processPrintElements(elements: any[]): Promise<boolean> {
    if (!this.nMPrintSocket) return false;
    console.log("elements", elements);
    for (const element of elements) {
        let res;
        switch (element.type) {
            case "text": // 文本打印
                res = await this.nMPrintSocket.DrawLabelText(element.json);
                break;
            case "qrCode": // 二维码打印
                res = await this.nMPrintSocket.DrawLabelQrCode(element.json);
                break;
            case "barCode": // 条形码打印
                res = await this.nMPrintSocket.DrawLabelBarCode(element.json);
                break;
            case "line": // 线条绘制
                res = await this.nMPrintSocket.DrawLabelLine(element.json);
                break;
            case "graph": // 图形绘制
                res = await this.nMPrintSocket.DrawLabelGraph(element.json);
                break;
            case "image": // 图像打印
                res = await this.nMPrintSocket.DrawLabelImage(element.json);
                break;
            default:
                console.error("Unsupported element type:", element.type);
                return false;
        }
        if (parseInt(JSON.parse(res.resultAck.errorCode)) !== 0) {
            console.error(`Failed to draw ${element.type}:`, res);
            return false;
        }
    }
    return true;
}
```

3.4.1 二维码绘制 DrawLabelQrCode

代码块

```
export default class NMPrintSocket {
    /**
     * 绘制二维码。
     *
     * @param {Object} json - 包含二维码信息的JSON对象。格式如下：
     * {
     *   "x": number, // x轴坐标，单位mm
     *   "y": number, // y轴坐标，单位mm
     *   "height": number, // 二维码高度，默认宽高一致
     *   "width": number, // 二维码宽度，单位mm
     *   "value": string, // 二维码内容
     *   "codeType": number, // 条码类型：
     *                     // 31: QR_CODE
     *                     // 32: PDF417
     *                     // 33: DATA_MATRIX
     *                     // 34: AZTEC
     *   "rotate": number, // 旋转角度，仅支持0、90、180、270
     * }
     *
     * @return {Promise} 返回一个 Promise，解析为绘制二维码的结果
     *
     * @description
     * 1. 绘制元素前，必须先初始化画板
     */
    public DrawLabelQrCode(json: String): Promise<any>
}
```

代码块

```
//返回数据示例
{
        "apiName": "DrawLableQrCode",
        "resultAck": {
                "errorCode": 0,
                "info": "draw qr code success!",
                "result": 0
        }
}

/**
 * 打印元素处理方法
 * 支持多种打印元素类型，按顺序执行绘制操作
 * @param elements - 打印元素数组
 * @returns 是否全部元素处理成功
 */
private async processPrintElements(elements: any[]): Promise<boolean> {
    if (!this.nMPrintSocket) return false;
    console.log("elements", elements);
    for (const element of elements) {
        let res;
        switch (element.type) {
            case "text": // 文本打印
                res = await this.nMPrintSocket.DrawLabelText(element.json);
                break;
            case "qrCode": // 二维码打印
                res = await this.nMPrintSocket.DrawLabelQrCode(element.json);
                break;
            case "barCode": // 条形码打印
                res = await this.nMPrintSocket.DrawLabelBarCode(element.json);
                break;
            case "line": // 线条绘制
                res = await this.nMPrintSocket.DrawLabelLine(element.json);
                break;
            case "graph": // 图形绘制
```

```
                res = await this.nMPrintSocket.DrawLabelGraph(element.json);
                break;
            case "image": // 图像打印
                res = await this.nMPrintSocket.DrawLabelImage(element.json);
                break;
            default:
                console.error("Unsupported element type:", element.type);
                return false;
        }
        if (parseInt(JSON.parse(res.resultAck.errorCode)) !== 0) {
            console.error(`Failed to draw ${element.type}:`, res);
            return false;
        }
    }
    return true;
}
```

3.4.2 二维码绘制 DrawLabelQrCode

代码块

```
export default class NMPrintSocket {
    /**
     * 绘制带logo的二维码。
     * @param {*} json - 包含二维码信息的JSON对象。格式如下：
     * {
     *   "x": number, // x轴坐标，单位mm
     *   "y": number, // y轴坐标，单位mm
     *   "height": number, // 二维码高度，默认宽高一致
     *   "width": number, // 二维码宽度，单位mm
     *   "value": string, // 二维码内容
     *   "codeType": number, // 条码类型：
     *                     // 31: QR_CODE
     *                     // 32: PDF417
     *                     // 33: DATA_MATRIX
     *                     // 34: AZTEC
     *   "rotate": number, // 旋转角度，仅支持0、90、180、270
     *  "correctLevel": 2,//纠错级别，取值范围1-4，默认2
     *  ""logoBase64": ": string,//logo的base64编码(不含数据头，如data:image/png;base64,)
     *  ""logoPosition": ": 0,//logo的位置，取值范围0-4，默认0:居中，3右下·
     *  "logoHeight": number,//logo高度，单位mm，默认10mm
     *    "logoScale": 0.25,//logo缩放比例，取值范围0-0.33，默认0.25
     * }
     */
    public DrawLabelQrCodeWithLogo(json: String): Promise<any>
}
```

3.5 线条绘制 DrawLabelLine

代码块

```
export default class NMPrintSocket {
    /**
     * 绘制线条。
     *
     * @param {Object} json - 包含线条信息的JSON对象。格式如下：
     * {
     *   "x": number, // x轴坐标，单位mm
     *   "y": number, // y轴坐标，单位mm
     *   "height": number, // 线高，单位mm
     *   "width": number, // 线宽，单位mm
     *   "lineType": number, // 线条类型：1:实线 2:虚线类型,虚实比例1:1
     *   "rotate": number, // 旋转角度，仅支持0、90、180、270
     *   "dashwidth": number // 线条为虚线宽度，【实线段长度，空线段长度】
     * }
     *
     * @return {Promise} 返回一个 Promise，解析为绘制线条的结果
     *
     * @description
     * 1. 绘制元素前，必须先初始化画板
     */
    public DrawLabelLine(json: String): Promise<any>
}
```

代码块

```
//返回数据示例
{
        "apiName": "DrawLableLine",
        "resultAck": {
                "errorCode": 0,
                "info": "draw line success!",
                "result": 0
        }
}

/**
 * 打印元素处理方法
 * 支持多种打印元素类型，按顺序执行绘制操作
 * @param elements - 打印元素数组
 * @returns 是否全部元素处理成功
 */
private async processPrintElements(elements: any[]): Promise<boolean> {
    if (!this.nMPrintSocket) return false;
    console.log("elements", elements);
    for (const element of elements) {
        let res;
        switch (element.type) {
            case "text": // 文本打印
                res = await this.nMPrintSocket.DrawLabelText(element.json);
                break;
            case "qrCode": // 二维码打印
                res = await this.nMPrintSocket.DrawLabelQrCode(element.json);
                break;
            case "barCode": // 条形码打印
                res = await this.nMPrintSocket.DrawLabelBarCode(element.json);
                break;
            case "line": // 线条绘制
                res = await this.nMPrintSocket.DrawLabelLine(element.json);
                break;
            case "graph": // 图形绘制
                res = await this.nMPrintSocket.DrawLabelGraph(element.json);
                break;
            case "image": // 图像打印
                res = await this.nMPrintSocket.DrawLabelImage(element.json);
                break;
            default:
                console.error("Unsupported element type:", element.type);
                return false;
        }
        if (parseInt(JSON.parse(res.resultAck.errorCode)) !== 0) {
            console.error(`Failed to draw ${element.type}:`, res);
            return false;
        }
    }
    return true;
}
```

3.6 绘制图形 DrawLabelGraph

代码块

```
export default class NMPrintSocket {
    /**
     * 绘制图形。
     *
     * @param {Object} json - 包含绘制图形信息的JSON对象。格式如下:
```

```
     * {
     *   "x": number, // x轴坐标，单位mm
     *   "y": number, // y轴坐标，单位mm
     *   "height": number, // 图形高度，单位mm
     *   "width": number, // 图形宽度，单位mm
     *   "rotate": number, // 旋转角度，仅支持0、90、180、270
     *   "cornerRadius": number, // 圆角半径，单位mm，暂不生效
     *   "lineWidth": number, // 线宽，单位mm
     *   "lineType": number, // 线条类型：1:实线 2:虚线类型,虚实比例1:1
     *   "graphType": number, // 图形类型：1:圆，2:椭圆，3:矩形 4:圆角矩形
     *   "dashwidth": number // 线条为虚线宽度，【实线段长度，空线段长度】
     * }
     *
     * @return {Promise} 返回一个 Promise，解析为绘制图形的结果
     *
     * @description
     * 1. 绘制元素前，必须先初始化画板
     */
    public DrawLabelGraph(json: String): Promise<any>
}
```

代码块

```
//返回数据示例
{
        "apiName": "DrawLableGraph",
        "resultAck": {
                "errorCode": 0,
                "info": "draw graph success!",
                "result": 0
        }
}

/**
 * 打印元素处理方法
 * 支持多种打印元素类型，按顺序执行绘制操作
 * @param elements - 打印元素数组
 * @returns 是否全部元素处理成功
 */
private async processPrintElements(elements: any[]): Promise<boolean> {
    if (!this.nMPrintSocket) return false;
    console.log("elements", elements);
    for (const element of elements) {
        let res;
        switch (element.type) {
            case "text": // 文本打印
```

```
                res = await this.nMPrintSocket.DrawLabelText(element.json);
                break;
            case "qrCode": // 二维码打印
                res = await this.nMPrintSocket.DrawLabelQrCode(element.json);
                break;
            case "barCode": // 条形码打印
                res = await this.nMPrintSocket.DrawLabelBarCode(element.json);
                break;
            case "line": // 线条绘制
                res = await this.nMPrintSocket.DrawLabelLine(element.json);
                break;
            case "graph": // 图形绘制
                res = await this.nMPrintSocket.DrawLabelGraph(element.json);
                break;
            case "image": // 图像打印
                res = await this.nMPrintSocket.DrawLabelImage(element.json);
                break;
            default:
                console.error("Unsupported element type:", element.type);
                return false;
        }
        if (parseInt(JSON.parse(res.resultAck.errorCode)) !== 0) {
            console.error(`Failed to draw ${element.type}:`, res);
            return false;
        }
    }
    return true;
}
```

3.7 绘制图像 DrawLabelImage

代码块

```
export default class NMPrintSocket {
    /**
     * 绘制图片。
     *
     * @param {Object} json - 包含绘制图片信息的JSON对象。格式如下：
     * {
     *   "x": number, // x轴坐标，单位mm
     *   "y": number, // y轴坐标，单位mm
     *   "height": number, // 图片高度，单位mm
     *   "width": number, // 图片宽度，单位mm
     *   "rotate": number, // 旋转角度，仅支持0、90、180、270
     *   "imageProcessingType": number, // 图像处理算法，默认0
     *   "imageProcessingValue": number, // 算法参数，默认127
     *   "imageData": number, // 图片base64数据，不含数据头
     *                     // 如原始数据为 “data:image/png;base64,iVBORw0KGgoAAAANSU”
     *                     // 传入的数据需要去除头部，数据为，“iVBORw0KGgoAAAANSU”
     * }
     * @return {Promise} 返回一个 Promise，解析为绘制图片的结果
     *
     * @description
     * 增加接口说明：
     * 1. 绘制元素前，必须先初始化画板
     */
    public DrawLabelImage(json: String): Promise<any>
}
```

代码块

```
//返回数据示例
{
        "apiName": "DrawLableImage",
        "resultAck": {
                "errorCode": 0,
                "info": "draw image success!",
                "result": 0
        }
}

/**
 * 打印元素处理方法
 * 支持多种打印元素类型，按顺序执行绘制操作
 * @param elements - 打印元素数组
 * @returns 是否全部元素处理成功
 */
private async processPrintElements(elements: any[]): Promise<boolean> {
    if (!this.nMPrintSocket) return false;
    console.log("elements", elements);
    for (const element of elements) {
        let res;
        switch (element.type) {
            case "text": // 文本打印
                res = await this.nMPrintSocket.DrawLabelText(element.json);
                break;
            case "qrCode": // 二维码打印
                res = await this.nMPrintSocket.DrawLabelQrCode(element.json);
                break;
            case "barCode": // 条形码打印
                res = await this.nMPrintSocket.DrawLabelBarCode(element.json);
                break;
            case "line": // 线条绘制
                res = await this.nMPrintSocket.DrawLabelLine(element.json);
                break;
            case "graph": // 图形绘制
                res = await this.nMPrintSocket.DrawLabelGraph(element.json);
                break;
            case "image": // 图像打印
                res = await this.nMPrintSocket.DrawLabelImage(element.json);
                break;
            default:
                console.error("Unsupported element type:", element.type);
                return false;
        }
        if (parseInt(JSON.parse(res.resultAck.errorCode)) !== 0) {
            console.error(`Failed to draw ${element.type}:`, res);
            return false;
        }
    }
    return true;
}
```

3.8 标签预览 generateImagePreviewImage

代码块

```
export default class NMPrintSocket {
    /**
     * 生成图像预览图像。
     *
     * @param {number} displayScale - 图像显示比例，表示 1mm 的点数，可调整预览图大小。
     *                               例如，200dpi 的打印机可设置为 8，300dpi 的打印机可设置为 11.81。
     *
     * @return {Promise} 返回一个 Promise，解析为生成图像预览图像的结果
     *
     * @description
     * 增加方法说明：
     * 1. 在调用此函数之前，必须确保图像数据已准备好，否则无法生成预览。
     */
    public generateImagePreviewImage(displayScale: Number): Promise<any>
}
```

代码块

```
//返回数据示例
{
        "apiName": "generateImagePreviewImage",
        "resultAck": {
                "errorCode": 0,
                "info": "{\n\t\"ImageData\" :
\"iVBORw0KGgoAAAANSUhEUgAAAZAAAADwCAIAAAChXqV1AAAgAElEQVR4AezBeaznd33f++fr8/2db
WbOeMYzY+MZb+DxgAk2lN5QGiVRC71SkyqtKhVVAZEAoUsQ6U1RK4SaRKJqdRM1EamKyI3C4pKNViit
qlQJ/aMNNA2JaEoANxhsg5fx2B4vs585y+/7eV7zTY80R/POVUa3FhrpPB5R2bVr167rQVR27dq163o
QlV27du26HkRl165du64HUdm1a9eu60FUdu3atet6EJVdu3btuh5EZdeuXbuuB1HZtWvXrutBVHbt2r
XrehCVXbt27boeRGXXrl27rgdR2bVr167rQVR27dq163oQlV27du26HkRl165du64HUbkWSaioVJJQU
akkoaJSSUJFpZKEikolCRWVShIqKpUkVFQqSaioVJJQUakkoaJSSUJFpZKEikolCRWVShIqKpUkVFQq
SaioVJJQUakkoaJSSUJFpZKEikolCRWVShIqKpUkVFQqUbkWSaioVJJQUakkoaJSSUJFpZKEikolCRW
VShIqKpUkVFQqSaioVJJQUakkoaJSSUJFpZKEikolCRWVShIqKpUkVFQqSaioVJJQUakkoaJSSUJFpZ
KEikolCRWVShIqKpUkVFQqUbkWSaioVJJQUakkoaJSSUJFpZKEikolCRWVShIqKpUkVFQqSaioVJJQU
akkoaJSSUJFpZKEikolCRWVShIqKpUkVFQqSaioVJJQUakkoaJSSUJFpZKEikolCRWVShIqKpUkVFQq
UbkWSaioVJJQUakkoaJSSUJFpZKEikolCRWVShIqKpUkVFQqSaioVJJQUakkoaJSSUJFpZKEikolCRW
VShIqKpUkVFQqSaioVJJQUakkoaJSSUJFpZKEikolCRWVShIqKpUkVFQqUbkWSaioVJJQUakkoaJSSU
JFpZKEikolCRWVShIqKpUkVFQqSaioVJJQUakkoaJSSUJFpZKEikolCRWVShIqKpUkVFQqSaioVJJQU
akkoaJSSUJFpZKEikolCRWVShIqKpUkVFQqUbkWSaioVJJQUakkoaJSSUJFpZKEikolCRWVShIqKpUk
VFQqSaioVJJQUakkoaJSSUJFpZKEikolCRWVShIqKpUkVFQqSaioVJJQUakkoaJSSUJFpZKEikolCRW
VShIqKpUkVFQqUbkWSaioVJJQUakkoaJSSUJFpZKEikolCRWVShIqKpUkVFQqSaioVJJQUakkoaJSSU
JFpZKEikolCRWVShIqKpUkVFQqSaioVJJQUakkoaJSSUJFpZKEikolCRWVShIqKpUkVFQqUbkWSaioV
JJQUakkoaJSSUJFpZKEikolCRWVShIqKpUkVFQqSaioVJJQUakkoaJSSUJFpZKEikolCRWVShIqKpUk
VFQqSaioVJJQUakkoaJSSUJFpZKEikolCRWVShIqKpUkVFQqUbkWSaioVJJQUakkoaJSSUJFpZKEiko
lCRWVShIqKpUkVFQqSaioVJJQUakkoaJSSUJFpZKEikolCRWVShIqKpUkVFQqSaioVJJQUakkoaJSSU
JFpZKEikolCRWVShIqKpUkVFQqUbkWSaioVJJQUakkoaJSSUJFpZKEikolCRWVShIqKpUkVFQqSahcv
rwxn2/O5/OtrS1Aba0BR44cofLCC88lCzCXEVsyQDftxgMHqZw9e5ZJEjUT9YYbbbqBy9uzZJCqQBEgC
7N+/n8rZs2eBJCqQhMkNN9xA5dy5c733JGomTg4ePEjl7JnTI7MdIHIONNx8ciNq1See+45rqAC6k0
33URFpZKEikolCRWVShIqKpUkVFQqSaioVJJQUakkoaJSicq1SEJFpZKEikolCRWVShIqKpUkVFQqSa
ioVJJQUakkoaKy0wgb5y/svWE/lbNnz6pMMmFywwL03ULl8+TLQJkDvfRgGYDaabUVHZpgJJ1NYaFRVQg
SRM1NYaFZVKEioqoAJJABVorVGZz+dJeu/qMAxA711dXFykcvHiRcBtgAocOHCAypgsvvKDyoiwwdtPN
XL3p8M1UVCpJqKhUklBRqSSholJJQkWlkoSKSiUJFZVKVK5FEioqlSRUVCpJqKhUklBRqSSholJJQkW
lkoSKSiUJFXXtt3/n3GvvWZ4Nw8rSA2ttWEYFhcXqaiTvvvvJOKyjY1CZMkVObzOd
B7d9J7d7Jv3z4qzz33nBPAbb33Y8eOUXnXKp/73O0d67/P5XJ3P530yn8//5t/8m
1Tuv//+PmGnd7/73VTuv//+JOrevXs3NjbR/nj27urp65MiR7//+vcn/lZWaFy8uRJ
4NixY+yUhIpKqVJFRUKkmoqFSSUFGpROVaJKGiUklCRaWShIpKJQkVlUoSKSiUJFZVKVK
oqFSSULl48WISFWitcDcOQycLCApXe++bmprq0tJTkYx/7RJLz58/Pjw8cPKtQHjTmjRUGSGttQsL
47kwcffJDKK17xNhyeeeebK1aubm7Ou3jaAwNDPr1f1eShqu+4+kkw4ZHS0c9e2+HvdDtLqLJAcLrFDHFGa
CeeeILvPTrDKeTeq/9nDQvssxe6/F7/k2NtZdE7nHNTzI67+K+/ngrWx3ZnjJ6eo/B8l1f/O42oxHsA
4Ni+Fx3fxZl39+3uZ/Sj72QKmfPnpT5w9KRG6IlPIAbGYRqnm8pKjFbEMMwVPsQQ6hxvaGGZs/pxbIU
CLwqlTp6jcsst6I+nyLgt2fyGTIgbEXyQbsFEvqZ6RmuoXfm+hEUkbRzqG/Z8GUSlHnLzXJwNJD9K6JHXFWbFzk4Z
ASwrdMxOaRJExKBzRoMqKbFPwNMWnK9uKnWmovc9zgJ3gymKHTFd8LzvnUIMbHaMY+Z2a5S7Ii+e6lCQfkiOLOL4
ALwB2k+xSqd5CKQiBqMowsr0gxtJBn+mGiKYDZ8i+4spqy89fd7ZkO3PNLx7VdVEmUhdZ2Cy5QpExGLh0E3Yx
ZkEZJFXHwmyM0ufNcM32kqKZOB6Q2jT6fHy6eYDmnTKhWhDqCeFjbzkV4Gnl0QzN8xbQjyjmbZz6zy4/b5mjLkDE5B
mHHGRlODqKdfcfpsJq9WTHv5UENbLdDnTCsqVvCkdt/fLHk6Kdu1BS1SnNvfKgB8T4m6bZhelbaZ9dVGZ0GRrMc5DG
wOnNjlZJimOaMc9WUsnLyPbjDePDsd1o0KbOrYHS6wWDP2Lbe2KrJ7ucVcDOsPWhVsRCbZvb0jrvGZgdf9g2uNl
GF3eUBy08aL9fAVV1UAXNr9FG5fwFjaPthKZI0zQyaRdsmv3ktjU3zOiE9ee6s3dOA7NJtwAqKfOHWF2W6VYWQZXpX
ATPqSqzTkRtu/d0W15CJqQMSyIHWeSIiAxBZjpjxcKPtI8Fr1q7cE9ifUqFWKTRv1Hc7kCF8qeNf5N7iH5d6Z8sCp1
ruzvKO31IT8BCd5b3qeyGsYNXd2Is0WHjQBVhGrFDN6jpcmC7L3sLeIXqYNQkr7V1cFjlbpPhmy9PTNeeI0Dbyk
fV1uPNG5Hm5bkKUyI5f0KkWZ4hE8kHmAdDGVYx5MuzcbnO2UAedtYhc5nBGtBtTtxAYkDCrdCsyLWkwtEgWHD1j0
N4FF35AJxMV0OSz8pLgm5XYlUZsXg4jmbxrEuimuu5N9oURC6CCOHcw1i0TqOAUAbW4L21/cpMcSt7m6YYsWDZp
qQFJEm0YB2xxd8eQeTm1sRrQI0KAKtr81DyOOKgQA4sOOh8qJqKjKSlwtMLbeEmzJpQkjSFNHE27qcgzMhabJaQ1
04bMpKF8SsVB6Jj4SZZuUk6SvjWsbzN7OhsTR0ik8p6itCLmfVtvWHPkzpgGw4OLCOQZNF16I7OdLbgRVK3NH6cMWqtfT9g
fEZNSxBqcJc3Rm5xRaoR3udNkdgnKNcRv7Pz1DJ2sMtPl6uBjklIwfx4uMbAeLWC6mp8RTanEdDmHeVPnM7d3P71kPJD9fN2pE4
zv17Fn4kZb5bfoQSkYaHZ3+SBG3mkkB4L+FfGCRBmS8gPA8wBCWIhymYoG8q0AtwXrYB9e22q6cDBqL4FBkcqhPKESLP3yGm7
CFKKMbVdEF4jlXVI5LBoIYmmT6WRJBTxS8YBcd3HXDqXQDGChJ1lMQNkaRRAVaEdRQnbTYXWFLdTQR0zyv73ixAyoCb3
hefXnYxlMFRBD2aAZAYA0ZWt8/mLXj8PFxRxsr6E4h5x8ugg7/AiHMe0QqEh+ZCWNIh7ExgCvRfA8AckR3g2tdNpJrtoAyJ
Rzu5Rx5tpOw2D8gYHh0Fm8dGhAQS3BVT2Y8GcOQ8z3hjjRuOzQ6A+J62Blrc7RtdfHK7DT0P5rlq7S2g/s+LcNGSZDGNBIC
iZ8yB8GOp1GiazsnxI+9SGKKMgr6kiHvsmYUqHq6FoHyi9kWR6BTuvrx5uYPh+Nt3HLuFh1CuLXSe4JDvgjTKr37fdG7TV2B4
Uc7zF9cGnxIbLfLG1IV7V1PrLCPSP9Rnk3Pf3iATqAWe+tHmjQaK9fsm1MBNcAfSh+3QNpo2cWpdJjwHK9g7PKQ6EBW1XFE
Zy5rvtt3pdC2bbzgNNQ+APdhGVsWz+gnekD2L9WFsNvaB6MHwSYJHI0nQZ4FxRfLLNZb+Nt68Hgi9f/DAWa+y8xgBqhgDHd
vBOWz1zkbW6uIE4bz6KNFKMLN6e3L1LS9ERWYzlMlUe8GHFYrvUtAjRKHXCEyWwQPtBZkq/0yEpmBlmH6G6MzLfz2Hi0P
9yFAqQMHLbnsIEPZvbRpxJPyq6sbbIIvadXBKkX3J3lWXNMycAUiFvrakIvgxSAtDbg6zSvR5LaYe+6QSB3xGzZgWM6
3ms9f9kzjZJqvxH3LTDEW0E2z20H3wmkr3v9uVy0VFtJhO3k0lBsjyX7MwkTjfJwHvxWcY9aHkfxn7oGvUiGWvoDbkUadB3f
iEQAQtt5l2mEsgYWBbtlvsdzRhjPqTFMaWQn7KsSnxF1wUCt3pNfAZ1HaMH7iw4X9VPWTLJQLV3emvI+DFlTrJfKmvOX
yPqM8SfCpZtWoOPp4Ga0yJzhoCUDXVFQ0dS4YlgG7Ux8sMwjS1KMGCZk5pS5RaKAyB5KQFBfMU6PZ3OyU0eWu2NSHZIe
tuxnTFRAZGJ3Xe5JoYjWH7q6n19pZVkUgBXZm1Ksv6cHXRbXtUU0M5cgN1eGH+Mc4MGsDfbKDn89+oNu3UzzsJz8mOSrnrMBg
6D76rPn3u1w2GHUeHwiKv3cPOKVZTmlpVaA95r+AaTbEqZxfhTXLwKTsWmJY3bRVOfo+ECKwFk7Pmxz5Y6jKMK3AjCGt2
z1HcT4QxGwHkJ4bVjXPL5lA8T+Jtv3BJIGnjzRZxdl+jXDNdrbqtzBmfoaDz3gJS9WWCIhMBKE1GZk3m3s4MxAzsH4S38G
YqMSmUmVHNtiZm3M0S6bWnqD9pnqGChrdDB6u3ip8eM6M7yaBg1CLTFZoXEUCiVAkKtQbAR79XCKOXy7Nv5aLrBpwkjb99kh
dmSn2XJRrKmLsBSaH73R0UGd5MHHtJnAmR2LNj0Zu12YDvv2F7eWpxrQEpR2xoQHeU68i02WsV6JUFhaovi7gGF8kWLhE
UcPj3m7FMKtOnAtASvXzsF3fKo2KKnCF3bqc8FSz9a9Yh6G3BtWoOSFlJMBtVuNkD8nvBvRd5ciYHjWwLVXKIV9IjOx2dU3O
c3t8X3Gt8M7QqmsZ9B5QAdTsvIH6xR4MSX3FHZv5GYtNv3H3r8P5THQkS5dLPx5Idl7nJ2Y3fbO3nBPeSHmlXLB8ueX2A3LTy
6D9QGbvTxB3VUXQ7Lks2IdzdjCsJzpBNuxK87gjjIgLsKsU6ImVm7bjX1VSuxTxcoTAhO65eS9lBaDM8Jp3wWn2UfAxF
9d3i+QtIBJjG6sNmFCw63lFWPZTCVL4FuyUaHIVKCsfXNSvO5+2s7yoEk+8fgOaG1zVNwSh1+yyHDdw4w5M8oHkv3Q3
rGF2Y2qhnGSFZ5CGVyWH7mEkZS3YHMJByBydbPjaYr7ZS9cJL9pl7SbdqjnPlLr3sd6M0MEqOQBV6AyNJvHZ3MZB8SQG0F9
TkoeADc1cMcrUMYnt8KjG8HRXL6uqBABk9MthX6xPgrQHHB0wqfAr9eEcySkE21qhN8dT6bsIoFRjJYdPXZEqZIVtVnT6w
b3XJAE31U7FEsy3OIBuU5QGeJ8p3Fhyr6ABmbB7LJxB99bBtnHjkf2PnkUVwsWeJR3CRc4t2HqvdgWGEqbCwuoALW8rS
eQUYqvKfBv4RuSr4MKQSJ6JIyhGXDCd9IPb2PfrhIMoyPsxuo8ZtykJwHKwIlwNvwiA5r9qWVf3gukzyBPtfbwVGPRrYftx8K
tVdnZOaHe/Ua9rqlaeqFxWyXK4iuFSOQAZtPIdq7dR8r3Bo5Oul9b0lWGLsAS7Vw53FqpHsvWWQ4u6eKx+C7fwfo6fk28
AkT2DW2pHNvyFgzi8yyK8pdKHLq2rjT8Q9eyMR7Xs3QF5EWFMjB3rb7ucGw2lpb5VaXbbgDqPDq2dE4PeNIPXWJX3cPrVRhp
RpSi5sG4Xmxr0x3TUJHzPxOWbqgKkMVODmmdbQbjHbEVXZLZjUmCMsZKwtTaRaSoqIRUJumHr3f5Ap7pZYdfpN9i7w7vXg
a5V8qb5Il6b8jIRnEgUuLxsrPpPB4SNDfzTcSQ7GE7IcCNHSCTtoPKtaVM5jYsdEkHmGNq5qG1Go8Jwe0pI4aqEbFMbUrNP
VYHttS8P2RAr9rSWUkUeBhjMtqTzutfL5npHR9x6hhD9sn6twAzHa9SqEVhYYyyLbTOV8kGH7ruW9p5rW9DbZ1bOU1N
XoPEb6hQ9NEHKU3SdJGSv8ftCQ5SXJ8Wr3gL0NS6ThRNv9ZgQ4TL0kN4e6UBRw07sOXq63BUQcgJlsVvFrlqsJBJGq
ulS5fW19dXV1cXFxfPnj178uRJKi972cvW19fHcWytAa01KEnJK4FIVXCOcM25rQxP8fpU8aMEkWnVnYI2RkvoWN
```

```
iTA2toalUOHDjFprWUCJHn66aep3HbbbUycAE6eeuopKocPH3bCNhU4c+YMlcOHDzNRmahJnn/+eSor
KytqEiAJkMnFixepqFSSUFGpJKGiUklCRaWShIpKJQkVlUoSKiqVqFyLJFRUKkmoqFSSUFGpJKGiUkl
CRaWShIpKJQkVlUoSKr3397///T/5kz+5urr62c9+9i/+xb/4oz/6oydPnvxP/+k/Ufme7/me3vvGxs
a5c+c2Njb27Nkzm81673/8x39MZXV1dXFxcRgGtqnAs88+S2X//v1uA1Qma2trVFZXV4EkQBK2nTt3j
srq6iqgsi2JevHiRSr79u1jkoQrXLhwgcr+/fu5QhI1yblz56gcOHAASOIEUIHz589T2bdvH5WLFy9S
+ehHP/pbv/VbP//zP3/rrbc6aa0BSaioVJJQUakkoaJSSUJFpZKEikolCRWVSlR2ffvM5/Nnn30WWF5
eHobhN3/zN7/85S//9E//NPBHf/RHt99++1/6S3/p0KFD8/l8bW3t3LlzeybDMLBTEvXy5cvnzp1bXl
4+cODAMAxMVKD3vrm5ef78+eeff35tbU2dzWaAylVUrkUSKip/OjUJExVIojJJoiZhogJJADUJOyXpv
QOZqExUtrXWVP4UKtuSACqTJGoSIAnghEkSKkm4SpLe+1/4C3/hM5/5zEc/+tHXve513/zmN7/whS98
6EMfYqImOXPmzMGDB9m1U1R2ffucevIJMsxms49//OOPP/74Rz7yka985957EhVQgd67ylWScIUkKjslUYEkbFOBJPyZqUmAJC
oTNQmgAkkAlUkSlaskUZMwUZPwZ5CEbWoStqmtNa6gJvEKgNpaG4YhiQQoSZgk4SpJuEISK5Srq4+9
NBDn/rUp17/+v/j137tV37u536Oyebm5rlz544cOcKuK0Rl17fJxceefGEpfX3z4x//+Hvf+9719fXb
b7/92LFjFy9e3LNnzxve8IZ77713ZWVlc3Pz6aefPnny5NGjRw8fPgyovXegT+bzee9/9/Pnzjz32mHr
s2LG9e/eq8/m89z5OLl68+PDDD/55JPr6+tAktaayv8mKpCE/9+SMFGTqEn4M3OSROUKKldQk6hAEp
U/gyTDMLTWAJUrJOEKSdgpCTslUVtrN91006c//Rs/vvffVv/+2/3Xs/c+bMoUOH2HWFq
Oz6Nnn69DMf+Cf/5BW33/mTm5ub//9GTP/mTm5ub//Sf/tOf+ZmfAXrvs9ksyd69e69d69e0+cOPGGN7xhfX19YWHh7rvv
XllZUcdxVMeJurW1tbm5tbm5ueerUqYceeujQoUO033nprkEnyn883J883888999wDDzzwzDPPPzOdzJklms1lrLQm
QBEgCJGGShCskYVVvvPQmg8qdQkzBRgSRMVLYlUZMwUZOoSdimJlGTsE3lKiqgAr13tjkBnAAq21T+bI
ZhaK0BKldIwIQJOyVhpyRhmxRqC7r13tjkBnAAq21T+bI
```

```
YEyGRjY4PKvn37mCRhkgQ4f/48lQMHDiRhogKbm5snT5688cYbeSmpXM+ici2SUFGpJKGiUklCRaWSh
IpKJQkVlUoSKiqVJFRUdprP5w899NCrX/1qKsMwLCwstNbU3rsTYGtri0prjSuoTFQqSaioVFprVHrv
VFprVHrvVFprVHrvVFprVHrvVIZhYFsSts3ncyoLCwtMkjBJAmxsbFBZXV1lkgmQyZkzZ6gcOnQIyLZ
PfOIT3//933/+/PkbbriBl5LK9Swq1yIJFZVKEioqlSRUVCpJqKhUklBRqSSholJJQkVlp7W1tccff/
yee+6h8vu///vnz5//vu/7vuXlZWAYhiSttTNnzlDZv3//OI7z+RxwW59QyYSr9N6ptNao9N6pDMNAZ
RxHKsMwUBnHkcpsNqMyn8+pLCwssFMSYHNzk8ri4iKTTIBM1tbWqBw4cABIAmTC5Pnnn6dy880O3n6
6adbaxsbG48//vgf/uEf/uAP/iAvJZXrWVSuRRIqKpUkVFQqSaioVJJQUakkoaJSSUJFpZKEispOFy5
cePrpp0+cOEHlda973ec///nl5eX/9t/+25kzZ376p3/661//epLTp09TOXDgQO99Pp8DSZiM47i+vk
6ltUal905lGAaukmQ+n1OZzWZU5vM5lYWFBSpbW1tUFhcXqWxublJZWlpiWxK2ra+vU1lZWQFaa2oma
mvtwoULVA4dOsSktZZEzeSZZ56hogKnTp3at2/fT/zET7z1rW9dXFwchuF1r3sdLyWV61lUrkUSKiqV
JFRUKkmoqFSSUFGpJKGiUklCRaWShIrKTmcmd911F5UHHnggSe/9fe9734c//OET/MefetvbnzjnzQ
zzTdTueGGG9RxHNUkbltfX6cyDANXSMJkPp9Tmc1mSbjK1tYWlYWFBbYlYdvm5iaVxcVFdkoCbGxsUF
leXqayvr5OZc+ePeyUBLh06RKVffv2AUmYZNvZs2ep3HzzzUySAK01JqdOnaLysz/7s1/60pf+wT/48
fl8M0lrbTabtdZe97rX8VJSuZ5F5VokoaJSSUJFpZKEikolCRWVShIqKpUkVFQqSaio7HT69OnLly/f
eeedVB566CFgHMckvffWu99Pp/fe++9VG688cbW2jiOvXc1iZMLFy5Qmc1mQBJ22traorKwsJAEULn
C1tYWlcXFRSqbm5tUlpaWqGxsbFBZWVnhKknW1tao7Nu3TwWSAEmYXLhwgcr+/fuZZAJk8sILLev6ev
RoJkBrLcmRI0f+83/+z3/v376fy3//7fx/HMVdpk9lstrC0eNfLX8GunaJyLZJQUakkoaJSSUJFpZKEi
kolCRWVShIqKpUkVFR2OnXq1Hw+v+OOO6g88cQTKtB7d5JkHMfjx49TUR977LEPPfOADv/ALv6DecMMN
wIkTJx5++GEqS0tLTpLwhc3NTSrLy8uAyrYYkwPr6OpWlpSW1tQaobNvZ2aJ0ByNvuAyrYkwPr6OpWlpSW1tQaobNvY2KCytLTEJAlXWF9fp7KysjKspK
EKyQBLl26RGV1dZWllnqPyR3/0R6997+9asffvjh97//R3/0R6oBUGRQD3rMPg5Bf9ysGg1QxDK8jkf
DJqxvGIh1xN3yLxnxK4jPoaYMhUGpp5KSPC0wHqYNgj8eRqZr0nN5QqnbKmEpMhaTvNLRM3Pw1MPwvAvBm
kVSyPTYWy6MN
EKyQBLl26RGV1dZWdkgDnz5+ncvDgQSYq0FrL5LnnnqPyR3/0R6997WvVz3/+869+9asffvjh97//A/
P55uc+9zkqf/iHf8i21lomrbUkwzAACwsLd911F7t2isq1SEJFpZKEikolCRWVShIqKpUkVFQqSaioV
JJQUdnpiccfF+644w4qTz31SiZpEPXr0KJVTp04NwwDM5/NbbrkF+Ot//a+YEf
+IHW2jAMKpPWWpJPf/rTVN72trcBrTWgbUvy0Y9+lMpP/dRPZWazhckwDLPZbGFhYTabbGhYTabvfvd76bye7/
3e6212Wy2sLAwmywsLAzDcMcdd1CZz+cqExXovasrKytUvvGNb4wTtU9U4L777qPy4IMPttaGdja2h
only9fvnDhwl/5K3+Fyjvf+c5//I//8cte9rKTJ0/OZjNga2sryX333Ufli1/8oportNaStNaGYQCSH
D9+nF07ReVaJKGiUklCRaWShIpKJQkVlUoSKiqVJFRUKkmoqOz0jW98Y2Fx+fbbjlE5c+YMNHUYht57
a20Yoq6urlJ5/vnnnx3EExnHsvQNqa+3o0aNUnnnnvuOSYqoCYBjhw5QuWZZ55JAiRRgdaaeuTIESqnT59moiZhkuTIkSNUnn32WZUrJJnP57
qoCYBjhw5QuWZZ55JAiRRgdaaeuTIESqnT59moiZhkuTIkSNUnn32WZUrJJnP57fccguVp59+urUGjO
PoBOi933bbbVS+9rWvWvJVEBJ0mAV73qVVS+8pWvMFGTtNaybRgGQL377rvZtVNUrkUSKiqVJFRUKkmoqFSSUFGpJKGi
FSSUFGpJKGiUklCRaWShIrKTg899NDKysptt91G5eLapSFNTTIMg8pkaWmJytrausEbUlWV/dSef75
55kkUZOowKFDh6g899xzQBLg8OHDQBLg8OHDVE6fP1ETQKoQJIjR45Qefb
ZZ1UmmKpMkN910E5Wnn35aYSrvPPPvvss1QOHTpE5a677qLywQcfAJKeJqr3/960nGcQ
RUQE0yDENr5EXw8lccZ9dOUbkWSaioVJJQUakkoaJSSUJFpZKEikolCRWVShIqKjt985uPLC0uHzy
irjxlprDbAFUJMAbVii0t2MqICaDE5mC0tU5lsbahIVaKICs+UVKhvra6213ruaBFBba4tLK1TW19eZ
qEASJ3v27KGytramMlGBJMDevXupXLp0SWgNUYO/evVQuXboEDMPQewfGcVTZ6qrq+ytbt
bGqAmAfbv30flzJkzKt/SAJXJoUMHqTz66KPQ/4QTIJCktZYEuPPld7Frp6hciyRUVCpJqKhUklBRqSSholJJQkVlp7W1tc
SholJJQkWlkoSKSiUJFZWdHv3mI7uLi8eOHaOyf/Pnz3//p/kY+e//e8Lxx/ucuv+ffuo1pkkQ0HGZ
m4kAdQkKpM2LFHp4wYTNYmaP9EWqYzzdRVoralt
w2yZyrixiSJkyTqbHkvlfXLFwx0uULvfe++A1SeOnWy7+SktZZETfLyVxxn105RuRZJqKhUklBRqSS
holJJQkWlkoSKSiUJFZVKEioqOz36zUcWlxaPHj1Kpezs7Oc7cWFxeP3Xo7lfn6pWEYSAxJ+BN50QIVF
AMJNDtXR1my1Tsm4CaRGXsmQ1A2iIV++Y4jq21+XzeWhuGofcODLNlKvOty621tGbvahKg9z5bWKEy3
7qcbpIeBtJ7twWYLaxQcWtdZZJEBdRhaQ+VvnlZBZIAKpNhaQ+VjfWLTdR5pAukO+K+1YNUTj35hNon
6jiOvXcgSWsNaK3dcecr2LVTVK5FEioqlSRUVCpJqKhUklBRqSSholJJQkVlkoSKyk6PfvORxcXFY7f
eTmW+fsmWWRoJLbwoAZIFKn7LmERNBuiA2tqMim7htyQhsfckahuWqPRxo/feWlN778MwDOos2Uq43
y9tab23ofZrI9jEnWYLVOZb10G1GEYeu9JAHW2sEKljxtM1CRqXtTNwjKVvnkZSNJ7b8Ng70za4gqVc
WNNba2pQO+doTUZlvZQ2Vy7AKjAiC9Kou7dd4DKU6dOqr33cRx+77o4jsDdJ17FS0nlehaVa5GEikolCRWVShIqKpUklFQqSaio
CRWVShIqKpUkVFQqSaioVJJQUakkoaKy08lHH+Y3fcSWW+finbbAGSkBctUNFRBZJwhWSgMo5bLfb
```

```
eW2tA7+RFrSUDlXG+3lpTe+9sSzLMlqn0cQNQW2skfRyBJllYpjLO15vYorbWeu9qe9GwRMWt9R4a6d
hao4sCWVimMs7Xm6hJfFFLE7UtrlDpm5fVJCMmcTKQYWkPlY31i036i8KLVKD3vm/1IJVTTz4BjOPYe
1d772rv/e4Tr+KlpHI9i8q1SEJFpZKEikolCRWVShIqKpUkVFQqSaioVJJQUdnp0Ue/sbgwHLv1Tirz
9UvZxtD4E3nRAhUdVbYlAXrvw7BAxT4Heu+JfEvTUZ0trFDp44baex+GQWWSpA1LVMb5OmNvwzDvY+W
39EQAACAASURBVJJhNptvbSWZLaxQGefrQJLee17UZdIWV6g43+jjCLTWSIB5H2dtyGyJilvrKttUJs
PSHip98zKgAgGSEQfSFleobF2+mKT3zmRu70h33+pBKqeefELtvY/jqPbe1d77iVfew0tJ5XoWlWuRh
IpKJQkVlUoSKiqVJFRUKkmoqFSSUFGpJKGistOj33xkcXHx2K23Uxk31lprvCgxvCiJ2oYlKjpyJb8F
aMMCFfscOjCOY7apbViiMs7XkwBqEibqMFumMt+6nKG10RHbbHDsQJI2LFEZ5+tAa42uvfeQri2zhRU
q863LTdTWmppEBdriCpWtzTVgEJIRgXSTDEt7qIwba6213nsStfc+m81678PSHirrly8ATXpQASd79x
2gcurJJ9Q+GcfRCXD87lfyUlK5nkXlWiSholJJQkWlkoSKSiUJFZVKEioqlSRUVCpJqKjs9Nij31hcX
Dx67DYq48Ya0IahYyZqXtQWqfQ+B5JARw1hUFubUenjhtpaG8cRSKImGWbLVOZbl4fZzN7Z5mS2sEJl
vnW5SQ9N1MyGdHsYZstU5luXgYEAPaSbRG2LK1Tsm31rztB670mAdJO0xRUqmxuXmgzD0HtPgsq3DEt
7qGxdvpgESKICakuG5b1Uti5fHBFQARVQ9+47QOWpUyf7VZIcv/uVvJRUrmdRuRZJqKhUklBRqSShol
JJQkWlkoSKSiUJFZVKEioqV1Aff+ybi4uzo8fuoNI3LwdIenhRa03Ni9oilS9/6YskaBIgE+A7XnMfl
f/5wJdba2prTU3Se09yz6tfQ+XrX/uqmqT3nonaWrv7xKuoPPT1B9UkY+9J0NYacPeJV1F5+KGvAa21
3nsSJ0mO3/1KKg99/cEkTNRMgLuOn6Dy0NcfHIZhPo6BJL33YTaz9+N3v5LKNx55CGitAWprLZPbbr+
TysULZ1prQBIVUIG9+w5QOfXkE2rvfRzH3rvaewfuPvEqXkoq17OoXIskVFQqSaioVJJQUakkoaJSSU
JFpZKEikolCRWVK1y6dGnv3r2nTp06duwYlXFjrbUG2AIkUV944YXDR26h8sY3vrG1NgzDfD5XoUNL8
vnPf57Kd37ndwIq4+KT3rn7pS1+icu+997bWgHEcl5eX19fXk6gPPPPAfHEcl5eX19fXk6gPPPAAlde//
/vUbGxtqkHaVo3KT3rn7pS1+icu+997bWgHEcl5eX19fXk6gPPPAAlde//
k8prXvMalUkSYDabbWxsfPWrX6Xy+te/Xh0nS0tLTpJ88YtfpPLa174WUJmoSYAvf/nLVN74xjf03od
h0Lyo/S989rP/lcr3fu93t9bUcRxba+rW1tab3/zmf/7P/28qJ594LMk4jn2aY/KtU
hCRaWShIpKJQkVlUoSKiqVJFRUKkmoqFSSUFG5wtbW5lOnnlxYWDh67DYq8/VLwzD0kERtw4DpvQ+zR
Sr33/9xJmPf6r0PbQFoA+/44XdT+djT9T/PA73kXllz95P+BkNpvN5/Pee2vth9/x
Lir/+v6PA22ytbWVRG2t/dAPv5PKJz7+0QkOTtP/QOKr/yy/+6984kiQr03t/xzh+h8iu
//K9ba1tbW7PZbBxHoLU2n8/f8c4fZi//OR2WwGbG5u/sZv/Mb6xk
ZrbRzH3/3d36Ny6skngPl8PpWIzfjs3ZVy6TUKZv+PcwnFwijPAL4o09FO9KQSKmXk0Z
...
```

```
QjnMHOvaBIzmVUVHNsph5ozr68VYGMdxRVljCcVwcCFzPL6ujrGUQrXWWkpRh6NLmTPeuJoEqMRJMRu
rC5czZ7xxVR1KqTpGNqobBxfvyZz19StJABVIoiZZXbicOeONqyqgMqm1AsPRpcxZX78CqJmoSYDVhc
uZc+3qy5k4SaLUWt903wOZ8/xzzyQZx7E2ahKglKLe/8D9Dz74cLrXQ802gMxRMwfIHDVzgMxRMwfIH
DVzgMxRMwfIHDVzgMxRc4uvPv3lg4NhWB0++ujbMufmjStA2TCVMElCOcwc680aCyuV1EC1YspwlDl1
vKEmqbUySQKU4Shz6s1rSSwU40aBapJyeDFzxhtXcwsVSDIcXcqc8cbVJGoppdYKZDIcXcqcmzeuDGH
DCeDk4OI9mbO+fiUJoGZSSlGHo0uZc3ztFSCJClTymsOjy5lz7erLSWwyqTVvuu+BzPna88+q6/Varb
U6qbWWSa31zQ899OYHHkz3eqjZBpA5auYAmaNmDpA5auYAmaNmDpA5auYAmaNmDpA5am7x3NNfYZVy4
ehtD/9Q5ow3rpZSKgGSAJlQDjPHutaRUqKZ1FpLKZTDzBnX14EkGGslCajl8GLmeHw9G3A8roEkQ3jV
wYXMWV+/UkqptWYC1FqHYSiHFzNnvHG1ErUYNpJaa1bD6uBi5ty4/gqQZgi1VmB14XLmjDeullKO6wi
k4FiHsFEOL2bOjeuvZDKESkq1FopZXbicOdeuvpxEkuprktRa33Tfg5nzteefrbdIUmvNBEjy8EMP33
v//eleDzXdD9xXn/7ywcGwWh2uDodSykAZNgKrYQhDKUkoJZBEAiSREJKSV9VsaBpJqmmATGqtmVCtZ
EMtG+bPDSVjVTMUCTAer7/zne889dS/eObZ5+PxQw899M4nHn/LW95yMKyGUMexDEMSYG1Vh1BKWa/X
maxWq3EcgSQqkKTWmqFQLVAVSKMCKpBEBVQgiQqoJCaVqAdlGMexlKIC0WPrASXJSIAkQ1BJqg7DEEh
hHMchBNZ1XFFqrUmASlSqpRR1jEmollKSjDEJ1QJVK9mgmskYM5GkulFrTVJrLWW1Xq+d1FqTjOOoju
PopDZqkn/5xx/PktTsMtR0P3BfffrLh4erYTgYDspAWa1WQyilrIbBpJTCa0pJItkAMpEQjEnIoIaah
GCtgAKmGceRiVprTcL3GMkLf/bif/vf/Nrf//v/1fHNmxnXpZQxAk6GkGSMSVaF0QzDwZ889/wn/49P
/vW/+teAJKwGN9ZjmYzjmAQ4ruOKopKYALVWoJRSawWSAGNMUgzJOqbgWFcUFVArGUISQK21AtFaeFV
1KOUYhxoSSqnkk5/85JNP/p8f/shH1sfHOmasbMQUnNREMwxDKauvfO/Urv/Vbv/Whv/HvP/zww2NMQh
XIxEk0MMZSCtUxZkJ1jEncICWotdYktVYYxnG0qRN1HEe1TtQ6SfLj73wiS1Kzy1DT/cB95ctPXbh4e
TUMZcVAGSZAaZKUUpIASYA0EoIxCSETJVRCNImaxAmvqW4A2QBr5WD1j/7Rb3zoaQx8arBZqrUMYx3G1
WtVak1SiHh8fHx4eqoCaRk3yqU/93uHR0Qf+4vtrrZk4YaICtVbAjQLVSoagllLGWEqptVJNwkRNAiS
ptQKVFENSFai1llKSWFCBYqKU8k/+6Sf/pR/7sUcfewz2UZY17PSZJaK5DESRL/HEnGcawTtTZqrTXJOI5qrTXJj7/ziSxJzS5DTfcD98UvfuGey
5dLKavVahgJgok4+fIK5MylCZJKSUgAXIryIavSiEJRgVUQM1ELSYQX5WE1VCP10CG8l9+9GMf/9hX/9hHXY9JKimG
xER9/Cf+lS8/9YX/XOOYJSeDP94ZAh4XDwhYb+Icqul/Ad/7/t+dS/eObZ5+PxQw899M4nHn/LW95yMKyGUMexDEMSYG1Vh1BKWa/X
zyf/Df/YN/UOtaLeZ7Di5e+vjHP/7Nb3zj0qVLq9Xq8PDw6OhgtTr88If/05s3byYBnAAWitkAysHq7/
5PPff7z/+6/82/XOhbZSAKMkSowxiTFqJVsqEmptSbRMYlaazacjOPopE7GcxSa1Vroz7+xLuzJDW7
DDXdD9wXv/iFeyInBoZhGmRFWQ1DYmCgFqIXXZALkBEmqQBoVE82kEoljLeZ7xrharT72sV/5K+5
87PjmTbWY16i11p/6rcK/c+tVklklp/6Wd/+D3/57z
gBkqjAGIvZKKU4GeMQKhmCmuSfvXL1yQcfv+kvJsVSR5NK2KgClahJinkNUEkxQCUbf/fv/
hef+MTfo8pqLUCSf7J//5P3/++9ih5IMw5AE+JW/94mPfPjDPdY//zP//f7/m0CmRSyQsv
/Nnv/u7v/pW/8rMZXHaMEs0E1CVCJEyATNUmt0TGJkqTWtZqUOnEyjm0tNUmtdb1e29Ra1cefeHWpa
Xoab7gXvmmacLHBwcAAcHQyllGIYVFBSilrFarJEwqYZIT1CSUEs2txlpJCfFVdQBD9VW1OpQhZGMov/
Ebv/FLv/TLx9evWLMBqEnKweqll5zKGHjm9eX6+Ph5QkFmosYQif/8JTX/zSv/j3fvZn11Y1k
1KKCtRaATUTNRuFoQZIUzcIkAZIUsz3jDGJmkIJxVRSzEYlG8Dn/ujzv/epT//tv/2LhQCYnJjPf72
b//2K6+8cjwZx/H4+Hi9XqQ44Ac/+IEP/OH/Tp/gvYEwUavgag0e1lvrajly/+wR/8Xz/zM3+J6qsKmahJVCATNYm
aBFBrrTDUWnVMoiRZr9fAOI/cO2qSOZqnVWtr+c5Eeeolfj78sS1QZ1HQ/cC9tfPvYFRgODw9GhjMuIgXzu3jSz
opwDAM6kEZQpRiNlgNadQkqBIVKpMlc4qSYSoozZYzqkEt5DVUk5RSpmt/5X/9e+lt/0/U4D
MMYqY6RoWBeozI5Pj7+6Ec/+tM/+zM/3b7mWM9h4h9t/O/fTB2OrQppKHZAQUa0oqKqeRWtdYhJAFe
CiRRgSQqoCYBktRah2FQ46O6fvvM3f+vg4OAv/+iv/o/2HinI4FvM8sv
61rJhKlGTAMWolbxGzQRQk6hTeSXLBHfQ4IhQxJbvJVjmhS1ToD1eg3UWtfrNBOktRax3F0kkR94l0/kSWp2WhWo6f7/8K
UvffFgtTo8PDw6OhgYztCQIGMN3KFzBLBT3kL1LW33d57dLUVMnYPtDMTxkXrj9FwDFVSw9TYbiG7Blvhmfpo+w6o4zbn
OXKlVrrhQsXjo6OhgrwcBtnrjDhVXjAxKTLFXAOiOK5c6EGyAyNnyM+UY6vFQE58T
pYwxE8lQQ3JsHYbhc5/73/f/8Le+9a3XTT4+vnjx4rNx4oULFwXwdGLC0ZZPgC3KW0oSFuKZJWDHXvEgv
KsWADJFGBWitQa4UkJPFVJKm1JlFrrWqdHWijuNYa1bj3ESE0CPPGun8iS10wy1GwDyLUvGE8/fSXh1KKKWdTYQCCJTVHEq
KAg4Nho0xWq1UpBapGvBnWhzJNMA3KBHRHc8y8uZ6Ds8kHyz93v+eiMBuh5ehOZajtEYaTbaVJj0o3hxkl5Mt8ahbNPq4kKBm9wL3WWnbQxSHBEfF0WBh1QMi
```

```
y+luXzP/Znzyne/DSS5fM/9OVNXXnmp1lpKUZMAKpDECZBEzRxAzQRQc4shHNeRiQqoSUlSa83EE2qt
6jiOap2o6/U6iVonSdRSiro6OLh48fLb3/72dCegZhtAlqTmjeRLX/riUMpqtSqlHBwMQCnl4OAgSSk
FKBOglKImATJR83pqJmqtNcmly/flTL3y3W+reb173/TmNC9/50Ugifqm+x7MnO++/GeAeu+b3pwz9f
J3XkwC1Fp5TTVJJSqgJgHUNICaCZBEzUQF1LyqJNGRSa01iQpDklprEie1VhWotaq1VnUcxzpRa61qn
eT/USmrCxcu/OiP/li6OajZBpAlqXmD+aPP/+HBwcHh4WEpZbValZLVagWsVqtSSpJhGJgkKaVk4gRI
oiaptQJJ1ETNxj33PpAz9Z2XXlABNQmQ5L77H0rz0re/lQnkvvvfkjnfeemFTO67/6GcqZe+/S0134e
ahIkKqPlzJalqGkBNSiY6JgGS1FphUDNxAqi11iRqrTWTegt1HMdaq5Naa5qqlHpwePHxH38i3feBmm
0AWZKaN6TPf/4PDw6Hw9XRMAyr1QostwDKBFCSmkRJKqAmUfKqmkSpdQ088OaHc6Ze+NY3mSQ1Sa0Ve
PChR9K8+MI3kgKoDz70cOa8+MKfZvLgQw/nTL3wrW8mAZOoeVVJo2NeVdKoSYAkar4PNUkpRa21AknG
cQSSqEm8BTCOYxIn46TWmkSttapJ1FjW403K4cVL5fLFe97+wz+S7vtDzTaALEnNG9U4jp/7w392dHR
0cHCQ5ODgoJQCDMOQZBgGoJSiMslETaICtVZArbUCSd76yGNpvvH1ryVRSym11iSlFDUJkESttQKllC
QqoD7y6NvS/Ok3v57Jw299NHO++Y0/AdQkb33ksTTf+PrXmDgB1EcefVuaP/3m1zNRmdQJkyQqoAJJa
q2A+uhjP5TmG1//mgqogFpKcZJbqEmA3KLWCqiAmjlqEhVQkzhJoiZRa61J1Fqrk1qrWmtVa61Jaq2Z
uD7m0uEFLj7x7nelux3UbAPIktS84T355GcoXjy6BKxWqyTDMNVUAEnQBo1k3f8hR9N88xXv5J
ETQOoQCZAEhXILX74HT+S5rlnv5oE+KG3vyNM2zzzzydhE
mtNZNSSpJxHJmogJNSipN3/IUfTfPsM09nojJxkkQFMlGBWiuTWiuQRgVUQE3CpNaqJlEBNQngnFprk
lqrTa3V1yQQcHR0dP3G+N6ffE+6O4OabQBZkpruFp/9/c3Q3SilDMPApJSSahEkmgBMgiVpKUd/9E/9q
mj/6/B8CaibqMAy11lJKJiqQRk1SSnn8iXeneeoLf6wC73z8XZnzhX/ZTo8zz8XWme+sIfAyqgZvL
Ox9+V5otP/fM0ak5QgUwANYn6+BPvTvPUF/5YTQKoSYBaayml1poT1CRqkUUFMnFSSqm1JgHUTJwAmT
hJU2tVk6hAErXXWal7F5Morxx/86Z9Ktz3UbAPIktR030et9Q8+85/p/P+/nvsXq+BYRiAJ
Oq/8YGfTvO7n/qdTn/10EkD9yfe+P82P8+85nvXruyXq+BYRiAJKUUoJSiJgGSA
Oq/8YGfTvO7n/qdTIDMAZIAahrgfe//qTRPfvbTapL3vu8vZs6Tn/10EkD9yfe+P82Tn/10EkAF1FLK
e/6196b5/Sc/A6iZqEAmKpBGzS1+8r3vT/PkZz+dBMgt1ExUQAUyUfP9qUmATNRMADWvByRRk5j4moz
jzVK9ds/Fb7/nfX813f83qOn2F5BGTQPkVGoaII2aCZA5auYAadRMgDRqGiCnUtMAadTsMiBLUrPLUN
PtLyCNmgbIqdQ0QBo1EyBz1MwB0qiZAGnUNEBOpaYB0qjZZUCWpGaXoabbX0AaNQ2QU6lpgDRqJkDmq
JkDpFEzAdKoaYCcSk0DpFGzy4AsSc0uQ023v4A0ahogp1LTAGnUnACkUdMAadScAKRR0wA5lZoGSKNm
lwFZkppdhppufwFp1DRATqWmAdKoOQFIo6YB0qg5AUijpgFyKjUNkEbNLgOyJDW7DDDXd/gLSqGmAnEp
NA6RRcwKQRk0DpFFzApBBGTQPkVGoaII2aXQZkSWp2GWq6/QWkUdMAOZZWaBkij5gQgjZoGSKPmBCCNmg
bIqdQ0QBo1uwzIktTsMtR0+wtIo6YBcio1DZBGzQnAR0+ws12wCyJDXd2QHSqGmAnEpNA6RRMwHSqGmAnEpNA6RR
MwHSqLkdII2aCZBGTQPkVGoaII2abn+hZhtAlqSmOztAGjUNkFOpaYA0aiZAGjUNkEbNLgOyJDW7DDDXd/gLSqGmAnEp
NA6RR0+0v1GwDyJLUdGcHSKPmrgBp1EyANGrmrgBp1EyANGrmAJmjZgKkUXNXgDRquv2Fmm0Amfm0AWZKa7uwAadTsMiBLUrPLUN
PtLyCNmgbIqdQ0QBo1EyBz1MwB0qiZAGnUNEBOpaYB0qjZZUCWpaYXQZ
kSWp2GWq6/QXkTKmZAGnUNEAaNQ2QRs0EyJlSs8uALEnNLkNN120PyN1S080BsiQ1uww1Xbc9IHdLTT
cHyJLU7DLUdN32gNwtNd0cIEtSs8tQ0+0vICeouQWQiZo5QE5Q0wCZo+Z2gJxKTQPkDX7AsiS1Owy1
HT7C8gcNRMgjZo5QE5Q0wCZo+ZUQG5HTQPkBDX7AsiS1Owy1GwDyJLUdGcHSyBw1EyCNmjlATLDTAJmj
5lRAbkdNA+QENd0bA2q2AWRJarqzA2SOmgmcQRs0cICeoaYDMUXMqILejpgFygprujdQBo1DZAT1DRAG
SqLljQBo1DZAT1DRAGjV3DEijpgHSqDkBSKOm21+o2QaQanQzg6QRs0dA9KoaYCcCoKYB0qi5Y0AaNQ
2QRs0JQBo13f5CzTaALElNd3aANGruGJBGTQPkBDUNkEbNHQPSqGmANGpOANKo6fYXarYBZElqurMDp
FFzx4A0ahogJ6hpgDRq7hiQRk0DpFFzApBGTbe/ULMNIEtS050dIIjahogp1LTAJmjpgFyx9Sc
CkijpntjQM02gCxJTXd2gDRqGiAnqGmANGo2QDRGTQPkCmAXIqNQ2
QOWoaIHdMzamANGr2BZAlqdllqOn2F5BGTQPkBDUNkFOpaYDMUdMAuWNqTgWkUbMvgCxJzS5DDpFGzQjAhrhMH
xRMwHSqLkdIKdS0wBp1JwApFHTAGnUnACkUdPtL9RsA8iS1OwyyzFBsnmIu72NSH
I2aBshEzT4CsiQ1uww13f4C8kfNBEij5naAnEpNA6RRcwKQRk0DpFFzApFHTAJmjpgHSqDkBSKOm21+o
AGnU3A6QRk0DpFFzApBGTQPkBDUNkEbNCUAaNQ2QRs0JQBo1DZAT1DRAGjW7DMiS1HNwYGjZo7BuSOqZkDpFEzAdKoaYDMUTMBMkfNLgOyJDW7DDXbALIkNd0ygDRqGiCnUtMAadScAKRR0wBp1EyAnEpNA6RRs8uALEnNLkNNGjZo7BuSOqZkDpFEzAdKoaYDMUTMBMkfNLgOyJDW7DDXbALIkNd0ygDRqGiCnUtMAadScAKRR0wBp1EyAnEpNA6RRs8uALEnNLkNN120PyN1S080BsiQ1uww1Xbc9IHdLTTcHyJLU7DLUdN32gNwtNd0cIEtSs8tQ0+0vICeouQWQiZo5QE5Q0wCZo+Z2gJxKTQPkDX7AsiS1Owy1HT7C8gcNRMgjZo5QE5Q0wCZo+Z2gJxKTQPkDX7AsiS1Owy1
```

```
ZmDpBGzQRIo6YBMkfNBMgcNd3+Qs02gCxJTbcMIBM12wByx9TMATJR0wBp1DRA5qiZAJmjpttfqNkGk
CWp6XYBkEbNHCCNmhOANGoaII2aBshETQOk3t2PMgAABxxJREFUUdPtL9RsA8iS1HS7AEijZg6QRs0J
QBo1DZBGTQNkoqYB0qjp9hdqtgFkSWq6XQCkUTMHSKPmBCCNmgZIo6YBMlHTAGnUdPsLNdsAsiQ13dk
BcqbUTIDMUXM7QO6KmgmQOWq6/YWabQBZkpru7AA5U2omQOaouR0gd0XNBMgcNd3+Qs02gCxJTXd2gJ
wpNRMgc9TcDpC7omYCZI6abn+hpttfQM6UmgmQOWpuB8hdUTMBMkfNLgOyJDW7DDXd/gLSqLkrQBo1J
wC5K2rmAGnUNEAmavYRkCWp2WWo6fYXkEbNXQHSqDkByF1RMwdIo6YBMlGzj4AsSc0uQ023v4A0au4K
kEbNCUDuipo5QBo1DZCJmn0EZElqdhlquv0FpFFzV4A0ak4AclfUzAHSqGmATNTsIyBLUrPLUNPtLyC
NmgbIqdQ0QBo1JwBp1DRAGjUNkBPU3DEgjZp9AWRJanYZarr9BaRR0wA5lZoGSKPmBCCNmgZIo6YBco
KaOwakUbMvgCxJzS5DTbe/gDRqGiCnUtMAadScAKRR0wBp1DRATlBzx4A0avYFkCWp2WWo2QaQJanpz
g6QRk0D5FRqGiCNmhOANGoaII2aBsgJau4YkEZN98aAmm0AWZKa7uwAadQ0QE6lpgHSqJkA2YaaUwGZ
o6YBcio13f5CzTaALElNd3aANGoaIKdS0wBp1EyAbEPNqYDMUdMAOZWabn+hZtAlqSmOztAGjUNkFO
paYA0aiZAtqHmVEDmqGmAnEpNt79Qsw0gS1LTnR0gjZoGyKnUNEAaNScAadTMAdKouStAJmoaII2abn+hZtAlqSmOztAGjUNkFOpaYA0ak4A0
qiZA6RRc1eATNQ0QBo13f5CzTaALElNd3aANGoaIKdS0wBp1JwApFEzB0ij5q4AmahpgDRquv2Fmm5/
AWnU3BUgjZoGyAlqbgfICWrmADlBzT4CsiQ1uww13f4C0qi5K0AaNRMgc9TcDpAT1MwBcozAHSqGmATJav
2GWq6/QWkUXNXgDRqJkDmqLkdICeomQPkBDX7CMiS1Owy1HT7C0ij5q4AadRMgMxRcztATlAzB8gJav
YRkCWp2WWo6fYXkDOlZgJkjpoGyBw1JwC5K2r2BZAlqdllqOn2F5AzpWYCZI6aBsgcNScAuStq9gWQJ
anZZajp9heQM6VmAmSOmgbIHDUnALkravYFkCWp2WWo6fYXkDOlZgJkjpoGyBw1JwC5K2r2BZAlqdll
qNkGkCWp6bqum4OabQBZkpqu67o5qNkGkCWp6bqum4OabQBZkpqu67o5qNkGkCWp6bqum4OabQBZkpq
u67o5qNkGkCWp6bqum4OabQBZkpqu67o5qOm67nwAsiQ1uww1XdedD0CWpGaXoabruvMByJLU7DLUdF
13PgBZkppdhpqu684HIEtSs8tQ03Xd+QBkSWp2GWq6rjsfgCxJzS5DTddd15wOQJanZZajpuu67o5qNm
G00AWZKa7uwAaZKa7ujZBpAlqem6rpuDmm0AWZKa7uu6OajZBpAlqem6rpuDmm0AWZKa7uu6OajZBpAl
qem6rpuDmm0AWZKa7uu6OajZBpAlqem6rpuDmm0AWZKa7uu6Oajpuu58ALIkNbsMNV3XnQ9AlqRml6Gm
67rzAciS1Owy1HRddz4AWZKaXYaarurOByBLUrPLUNN13fkAZElqdhlqum67nwAsiQ1uww13dddz4AWZKa
XYaaruvOByBLUrPLUNN13fkAZElqdhlqum67nwAsiQ1uww13dddz4AWZKaXYaarruvOByBLUrPLUNN13f
kAZElqdhlquq47H4AsSc0uQ03XdecDkCWp2WWo2QaQJanpuq6bg5ptAFmSmq7rujmo2QaQJanpuq6bg5
ptAFmSmq7rujmo2QaQJanpuq6bg5ptAFmSmq7rujmo2QaQJanpuq6bg5ptA
FmSmq7rujmo6brufACyJDW7DDVd150PQJakZpehpuu684HIktTsMtR0XXc+AFmSml2Gmq7rzgcgS1Kz
y1DTdd35AGRJanYZarquOx+ALEnNLkNN13XnA5AlqdllqNkGkCWp6bqum4OabQBZkpqu67o5qNkGkCW
p6bqum4OabQBZkpqu67o5qNkGkCWp6bqum4OabQBZkpqu67o5qNkGkCWp6bqum4OabQBZkpqu67o5qO
m67nwAsiQ1uww1XdedD0CWpGaXoabruvMByJLU7DLUdF13PgBZkppdhpqu684HIEtSs8tQ03Xd+QBkS
Wp2GWq6rjsfgCxJzS5DTddd15wOQJanZZajpuu67o5qNmG00AWZKa7uwAaZKa7ujZBpAlqem6rpuDmm0A
WZKaruu6OajZBpAlqem6rpuDmm0AWZKaruu6OajZBpAlqem6rpuDmm0AWZKaruu6Oajpuu58ALIkNbs
MNV3XnQ9AlqRml6Gm67rzAciS1Owy1HRddz4AWZKaXYaarurOByBLUrPLUNN13fkAZElqdhlquq47H4
AsSc0uQ03XdecDkCWp2WX/N3y+mOLHwCY+AAAAAElFTkSuQmCC\",\n\t\"errorCode\" :
0,\n\t\"errorInfo\" : \"No error!\"\n}\n",
                "result": 0
        }
}
```

```
/**
 * 生成打印预览图像
 * 通过API调用生成打印预览图像
 * @param printData - 打印数据对象
 */
public async handlePreview(printData: any): Promise<void> {
    if (!this.printSocketOpen || !this.nMPrintSocket) {
        return alert("打印服务未开启");
    }
    if (!this.initBool) {
        return alert("SDK未初始化");
    }

    try {
        const initCanvasRes = await this.initCanvas(printData.InitDrawingBoardParam)
        if (!initCanvasRes) {
            alert("初始化画布失败");
            return;
        }
        const elementsProcessed = await this.processPrintElements(printData.elements);
        if (!elementsProcessed) {
            alert("处理打印预览元素失败");
            return;
        }

        const previewRes = await this.nMPrintSocket.generateImagePreviewImage(8);
        if (previewRes.resultAck.errorCode === 0 && previewRes.resultAck.info) {
            const imageData = JSON.parse(previewRes.resultAck.info).ImageData
            this.previewImage = "data:image/png;base64," + imageData;
            // alert("预览图已生成");
            console.log("预览图已生成");
        } else {
            this.previewImage = null;
            alert("获取预览图失败");
        }
    } catch (err) {
        console.error(err);
        this.previewImage = null;
        alert("打印预览异常");
    }
    this._updateReactState();
}
```

四、打印接口说明

4.1 设置打印回调

代码块

```
 export default class NMPrintSocket {
    // 添加打印回调
    public addPrintListener(callback: (msg: any) => void): (msg: any) => void
  }
```

代码块

```
let printListener = null;

printListener = this.nMPrintSocket.addPrintListener(async (msg) => {
    const resultAck = msg?.resultAck;

    if (resultAck?.errorCode === 0 && resultAck?.info === "commitJob ok!") {
        await strategyFactory.handleCommitSuccess();
    }
    //已接入历史版本客户仍可以使用printQuantity和onPrintPageCompleted字段获取打印进度

    if (resultAck?.printCopies != null && resultAck?.printPages != null) {
        strategyFactory.handleProgressUpdate(resultAck);
    }

    if (resultAck?.printCopies === printQuantity &&
        resultAck?.printPages === list.length) {
        await strategyFactory.handleCompletion();
    }

    if (resultAck?.errorCode !== 0) {
        strategyFactory.handleError(msg);
    }
});
```

4.2 移除打印回调

代码块

```
 export default class NMPrintSocket {
    // 移除打印回调
    public removePrintListener(callback: (msg: any) => void): void
 }
```

代码块

```
const cleanupListener = () => {
    if (printListener && this.nMPrintSocket) {
        this.nMPrintSocket.removePrintListener(printListener);
        printListener = null;
    }
};
```

4.3 开始打印

代码块

```
export default class NMPrintSocket {
    /**
     * 开始一个打印任务。
     *
     * @param {number} printDensity - 打印浓度，根据不同打印机型号取值范围不同，具体如下：
     *                                        - B3S、B203、B1、K3、K3W、M2：取值范围 1~5，默认为 3。
     *                                          - B50、B11、B50W、B32、Z401：取值范围 1~15，默认为 8。
     * @param {number} printLabelType - 纸张类型，可选值：
     *                                     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
     * @param {number} printMode - 打印模式，可选值：
     *                                  1：热敏
     *                                  2：热转印
     *                                  注意，不同打印机型号支持的打印模式有限制，具体如下：
     *                                  - D11、D101、D110、H10、B16、B18、B3S、B203、B1、K3、K3W、B11 仅支持热敏。
     *                                  - B50、B50W、B32、Z401、M2 仅支持热转印。
     * @param {number} count - 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）。
     * @return {Promise} - 返回一个 Promise，解析为开始打印任务的结果
     * @example
     * //返回数据示例
     * {
     *     "apiName": "startJob",
     *     "resultAck": {
     *         "errorCode": 0,
     *         "info": "startJob ok!",
     *         "result": 0
     *     }
     * }
     * @description 返回结果中的 errorCode 含义如下:
     *              - 0: 成功
     *              - -1: 失败，info 表示原因
     *              - -2: 打印机忙碌，info 表示原因
     *              - -3: 打印机接收到不支持的参数，主要是浓度、纸张类型、打印模式，info 表示具体原因
     */
    public startJob(
        printDensity: number,
        printLabelType: number,
        printMode: number,
        count: number
    ): Promise<any>
}
```

代码块

```
//返回数据示例
{
        "apiName": "startJob",
        "resultAck": {
                "errorCode": 0,
                "info": "startJob ok!",
                "result": 0
        }
}

const startRes = await this.nMPrintSocket.startJob(
    this.density,
    this.label_type,
    this.print_mode,
    list.length * printQuantity
);

if (startRes.resultAck.errorCode !== 0) {
    cleanupListener();
}
```

4.4 提交打印任务

代码块

```
export default class NMPrintSocket {
    /**
     * 提交一个打印任务。
     *
     * @param {string} [printData=null] - 打印数据的 JSON 字符串。
     * @param {string} printerImageProcessingInfo - 打印机图像处理信息的 JSON 字符串，包含打印份数信息，格式如下：
     * {
     *   "printerImageProcessingInfo": {
     *     "printQuantity": 1 // 用于指定当前页的打印份数。例如，如果需要打印3页，第一页打印3份，第二页打印2份，第三页打印5份，则在3次提交数据时，printerImageProcessingInfo 中的 "printQuantity" 值分别应为 3，2，5。
     *   }
     * }
     * @return {Promise} - 返回一个 Promise，解析为提交打印任务的结果
     */
    public commitJob(printData: string | undefined, printerImageProcessingInfo: string): Promise<any>
}
```

代码块

```
//数据提交成功返回数据示例
{
        "apiName": "commitJob",
        "resultAck": {
                "errorCode": 0,
                "info": "commitJob ok!",
                "result": 0
        }
}

//打印进度返回示例1：此回调的含义为第一页第一份打印完成
{
        "apiName": "commitJob",
        "resultAck": {
                "errorCode": 0,
                "info": "",
                "onPrintEPCCodeCompleted": "",
                "onPrintPageCompleted": 1,//打印完成份数回调
                "onPrintPageLengthCompleted": "38.00",
                "printQuantity": 1//打印完成页数回调
        }
}

//打印进度返回示例1：此回调的含义为第一页第二份打印完成
{
        "apiName": "commitJob",
        "resultAck": {
                "errorCode": 0,
                "info": "",
                "onPrintEPCCodeCompleted": "",
                "onPrintPageCompleted": 2,//打印完成份数回调
                "onPrintPageLengthCompleted": "38.00",
                "printQuantity": 1//打印完成页数回调
        }
}

//打印进度返回示例1: 此回调的含义为第二页第一份打印完成
{
        "apiName": "commitJob",
        "resultAck": {
                "errorCode": 0,
                "info": "",
                "onPrintEPCCodeCompleted": "",
                "onPrintPageCompleted": 1,//打印完成份数回调
                "onPrintPageLengthCompleted": "38.00",
                "printQuantity": 2//打印完成页数回调
        }
}


public async commitPrintJob() {
    if (!this.printSocketOpen || !this.nMPrintSocket) {
        return alert("打印服务未开启");
    }
    this.nMPrintSocket.commitJob(undefined, JSON.stringify(this.jsonObj));
}
```

4.5 结束打印任务

代码块

```
export default class NMPrintSocket {
    /**
     * 结束一个打印任务。
     *
     * @return {Promise} - 返回一个 Promise, 解析为结束打印任务的结果
     */
    public endJob(): Promise<any>
}
```

代码块

```
//返回数据示例
```

```
{
        "apiName": "endJob",
        "resultAck": {
                "errorCode": 0,
                "info": "endJob ok!",
                "result": 0
        }
}

if (!this.nMPrintSocket) return;
const endRes = await this.nMPrintSocket.endJob();
if (endRes.resultAck.errorCode === 0) {
    console.log("打印完成");
}

cleanupListener();
```

4.6 取消打印任务

代码块

```
export default class NMPrintSocket {
    /**
     * 取消当前的打印任务，并执行回调函数。
     *
     * @return {Promise} 返回一个 Promise，解析为取消打印任务的结果
     */
    public cancelJob(): Promise<any>
}
```

代码块

```
try {
const cancelJobRes = await this.nMPrintSocket.cancelJob();
if (cancelJobRes.resultAck.errorCode == 0) {
  console.log("取消打印成功");
}
} catch (err) {
    console.error(err);
}
```

五、回调说明

代码块

```

/**
 * {
 *   "apiName": string, // 调用的 API 名称
 *   "resultAck": {
 *     "errorCode": number, // 错误代码，0 表示成功，其他值表示错误
 *     "info": string, // 信息字符串，描述操作结果
 *     "result": number // 结果代码，通常与 errorCode 一致
 *   }
 * }
 */
{
  "apiName": "commitJob",
  "resultAck": {
    "errorCode": 0,
    "info": "commitJob ok!",
    "result": 0
  }
}
```

六、错误码相关说明

6.1 错误码说明描述

代码块

```
* 0-无错误
//打印机返回部分
* 1-盒子打开
* 2-缺纸
* 3-电量不足
* 4-电池异常
* 5-手动停止
* 6-数据错误
* 7-温度过高
* 8-走纸异常
* 9-正在打印
* 10-未检测到打印头
* 11-环境温度过低
* 12-打印头松动
* 13-未检测到碳带
* 14-不匹配的耗材
* 15-用完的碳带
* 16-不支持的纸张类型
* 17-设置纸张类型失败
* 18-设置打印模式失败
* 19-设置浓度失败
* 20-写入rfid失败
* 21-边距参数错误
* 22-超时错误
* 23-断开连接
* 24-画板参数设置错误
* 25-旋转角度参数错误
* 26-json参数错误
* 27-出纸异常（关闭上盖检测）
* 28-检查纸张类型
* 29-碳带与打印模式不匹配
* 30-设置浓度不支持
* 31-不支持的打印模式
* 32-标签材质设置异常，请重新设置
* 33-不支持该标签材质，请更换或重新设置
* 34-不支持RFID写入
* 50-非法标签
* 51-非法碳带和标签

//内部使用
//E_UNKNOW_ERROR = 255,
```