1. 简介

联想图像提供了打印机移动端解决方案，即LeSDK，以方便移动端开发基于联想打印机的应用程序。
LeSDK提供了打印机配网、发现、打印、扫描、复印，以及OTA等基本核心功能。

适用人员

- Android应用开发工程师和测试人员
- 产品经理

2. SDK集成

2.1 使用申请

LeSDK需申请授权才能使用，否则初始化会失败并打印如下错误日志输出:

```
Your package `your.package.name` is not supported by the SDK
Contact your LeSDK provider for permissions.
```

2.2 添加依赖

SDK包括如下两个库文件，将该库文件添加至`libs`文件夹并声明如下依赖即可

- `LeCommon-release-1.0.0.aar`，基础库文件，必选。
- `LeSDK-release-3.2.4.aar`，核心功能库文件，必须。

```
implementation fileTree(dir: "libs", include: ["*.jar", "*.aar"])
```

2. 打印机配网

打印机配网指将打印机连接到指定的无线Wi-Fi网络，目前仅支持2.4G Wi-Fi网络。

2.1 蓝牙配网

详见蓝牙配网文档`app_notes_ble_setup_v1.2.pdf`

3. 打印机发现

LeSDK基于SNMP协议发现本地局域网内的已配网打印机。

接口说明

```
LeSnmpDeviceDiscoverManager.discoverDevice(
        timeoutInSeconds: Int = 3,
        maxRetries: Int = 1,
        matchers: List<LePrinterMatcher> = emptyList(),
        callback: LeDeviceDiscoverCallback,
)
```

####参数说明

- `timeoutInSeconds`, 设备发现超时时间，默认为3秒。
- `maxRetries`, 搜索重试次数，默认为1次。
- `matchers`, 搜索结果过滤器列表，用于只返回满足特定条件的打印机。
- `callback` , 搜索结果回调接口。

其中，回调接口`LeDeviceDiscoverCallback`定义如下

```
interface LeDeviceDiscoverCallback {
    fun onDeviceDiscovered(printer: LePrinter)
    fun onDeviceDiscoverDone(printers: List<LePrinter>)
    fun onDeviceDiscoverError(error: LeDeviceDiscoverError)
}
```

各回调函数定义如下

- `onDeviceDiscovered`，搜索到新的设备`printer`，包含了设备名称与对应IP地址。
- `onDeviceDiscoverDone`, 设备搜索结束，全部搜索结果为`printers`。
- `onDeviceDiscoverError`, 设备搜索失败，错误原因如`error`描述。

LeDeviceDiscoverError

错误类别	错误原因
NetworkError	网络错误
TimeoutError	超时且没有发现任务打印机

注意： `discoverDevice`返回对设备是所有支持SNMP协议的设备，往往需要借助`matchers`参数过滤出目标设备。

4. 打印机状态查询

####接口说明
调用`LePrinter`类提供的`status`方法可查询打印机总体工作状态，如打印中、休眠中等。

```
```

```
fun status(callback: (statusClass:StatusClass, status: Status) -> Unit)
```

参数说明

- `callback`，状态结果查询结果接受回调函数，返回打印机状态类别`statusClass`及具体的状态值`status`。

此外`LePrinter`还提供了`isIdle`方法以判断打印机是否处于空闲状态。
(*打印机只有处于空闲状态时才可以进行打印、扫码、复印等任务*)

```
fun isIdle(callback:(result:Boolean)->Unit)
```

5. 打印

####接口说明
文档打印调用`LePrinter`类提供的`doJob`方法，接收一个`LePrintJob`对象作为参数。

```
fun doJob(job: LePrintJob)
```

`LePrintJob`定义如下：

```
class LePrintJob(
    var paperSize: LePrintOptions.PaperSize,
    var paperType: LePrintOptions.PaperType,
    var duplex: LePrintOptions.Duplex,
    var responder: LePrintResponder
)
```

参数说明

- `paperSize`, 打印纸张尺寸。
- `paperSize`, 打印纸张类别。
- `duplex`，长边或短边打印选项。
- `responder`，打印回调函数，用于输入打印文件和输出打印结果。

此外，其它可选打印参数有：

- `scale`，缩放模式，默认为`LePrintOptions.Scale.FitToPrintArea`。
- `copies4NoCollate`，多页打印，默认为1。
- `tonerSave`，省墨模式打印，默认为不省墨模式。

其中，打印回调函数`responder`定义如下：

```
interface LePrintResponder: LeResponder {
    fun pageFile(pageNum:Int, jobId:Int):String?
    fun jobEnd(result: LePrintResult?, jobId: Int)
}
```

各回调函数说明如下：

- pageFile，指定第pageNum页打印文件路径，图片格式。若返回空则结束当前编号为jobId的打印任务。
- jobEnd， 打印结束，结果如result描述。有Succeed和Failed两种结果，分别对应打印成功和打印失败。

注：目前LeSDK只支持图片格式打印，非图片格式文件需转换为图片再打印。

- .png，.bmp，.jpeg, 可直接打印。
- *.txt, .pdf, .doc, .docx, .xls, .xlsx, .ppt, .pptx，需转换为图片格式再进行打印，如WPS Office。*
- *.pdf, 需转换为图片格式再进行打印。*

示例代码

```
LePrintJob printJob = new LePrintJob(
        LePrintOptions.PaperSize.Custom,
        LePrintOptions.PaperType.Recycled,
        LePrintOptions.Duplex.None,
        new LePrintResponder() {

            @Override
            public String pageFile(int pageNum, int jobId) {
                if (pageNum > 1) {
                    return null;
                }
                return printingFilePath;
            }

            @Override
            public void jobEnd(@Nullable LePrintResult result, int jobId)
{
                //
            }
        });

LePrinter printer = new LePrinter("LePrinter", "192.168.108.104");
printer.doJob(printJob);
```

6. 复印

文档复印调用LePrinter类提供的doJob方法，接收一个LeCopyJob对象作为参数。

```
fun doJob(job: LeCopyJob)
```

LeCopyJob定义如下：

```
class LeCopyJob(
    var density: Int,
    var copies: Int,
    var sourceSize: LeCopyOptions.PaperSize,
    var targetSize: LeCopyOptions.PaperSize,
    var dpi: LeCopyOptions.DPI
)
```

参数说明

- density, 复印浓度，可选范围为[0, 6]，默认为0。
- copies, 复印页数。
- sourceSize，源纸张尺寸。
- targetSize，目标纸张尺寸。
- dpi, 像素密度。

此外，其它可选复印参数有：

- duplex，长短边复印模式，默认为LeCopyOptions.Duplex.Off。
- idcardType，身份证复印位置选项，默认为LeCopyOptions.IDCardType.A4Center。
- nUp，N和1复印，默认为LeCopyOptions.nUp.up1。
- paperType, 纸张类别选项，默认为LeCopyOptions.PaperType.Plain。
- mode，复印模式，默认为LeCopyOptions.Mode.Photo。
- jobEnd, 复印结果回调函数。

示例代码(Kotlin)

以身份证复印为例，相关参数设置和调用如下：

```
val copyJob = LeCopyJob(1, 1,
    LeCopyOptions.PaperSize.A4,
    LeCopyOptions.PaperSize.A4,
    LeCopyOptions.DPI.dpi300
)
copyJob.nUp = LeCopyOptions.nUp.IDCard
copyJob.idcardType = LeCopyOptions.IDCardType.A4Center
copyJob.jobEnd = { Log.d(logTag, "copy result: $it") }
printer.doJob(copyJob)
```

7. 扫描

文档扫描调用LePrinter类提供的doJob方法，接收一个LeScanJob对象作为参数。

```
fun doJob(job: LeScanJob)
```

LeScanJob定义如下：

```
class LeScanJob(
    val sourceType: LeScanOptions.ScanSource,
    val dpi: LeScanOptions.DPI,
    val colorMode: LeScanOptions.ColorMode,
    val paperSize:LeScanOptions.PaperSize,
    val callback: LeScanResponder
)
```

参数说明

- sourceType, 扫描类别，分平板扫描和自动进纸两种。默认为平板扫描，自动进纸只支持特定机型。
- dpi, 扫码像素密度，可选值见LeScanOptions.DPI定义，默认为dpi600。
- colorMode，色彩模式，如彩色扫描与黑白扫描。
- paperSize，扫码纸张尺寸。
- callback，扫码结果回调接口，定义见LeScanResponder。

此外，其它可选复印参数有：

- pathUsedtoStoreImage，扫描文件存放路径。

LeScanResponder定义如下：

```
interface LeScanResponder: LeResponder {
    fun progress(file: String?, pagePercent: Int, jobId: Int)
    fun jobEnd(result: LeScanResult?, jobId:Int)
}
```

各回调借口说明

- progress，扫描进度更新回调函数，用于更新当前扫描文件file的当前进度pagePercent。
- jobEnd，扫码结束回到函数，扫描结果为result。

扫描结果LeScanResult值:

结果	说明
ADF_PaperEmpty	ADF扫描没有纸张放入.

结果	说明
ADF_JAM.	ADF扫描卡纸
ADF_CoverOpen	ADF扫描过程中开盖
Succeed	扫描成功
Printer_Error	内部扫描错误
Cancel	扫描取消，包括App内取消和面板按键取消
Busy	打印机忙，稍后重试。

示例代码(Java)

如下是平板黑白扫描并动态展示扫描图像的示例代码

```
LeScanJob scanJob = new LeScanJob(
    LeScanOptions.ScanSource.Flatbed,
    LeScanOptions.DPI.dpi600,
    LeScanOptions.ColorMode.Gray,
    LeScanOptions.PaperSize.A4,
    new LeScanResponder() {

        @Override
        public void progress(@Nullable String file, int pagePercent, int
jobId) {
            Log.d(TAG, String.format(Locale.US, "scan file: %s, progress:
%d", file, pagePercent));
            if (pagePercent % 5 == 0) {
                Bitmap image = decodeSampledBitmapFromResource(file);
                requireActivity().runOnUiThread(() ->
scanPreview.setImageBitmap(image));
            }
        }

        @Override
        public void jobEnd(@Nullable LeScanResult result, int jobId) {
            String resultMessage = (result != null) ? result.toString() :
"";
            Log.d(TAG, "scan job end with result:" + resultMessage);
            requireActivity().runOnUiThread(() -> {
                scanStatus.setText(String.format("Scan Finished: %s",
resultMessage));
            });
        }
   });

String path = LeSDK.scanFilesDir(requireContext());
Log.d(TAG, "scan file path: " + path);
scanJob.setPathUsedtoStoreImage(path);
printer.doJob(scanJob);
```