Gopenbanking

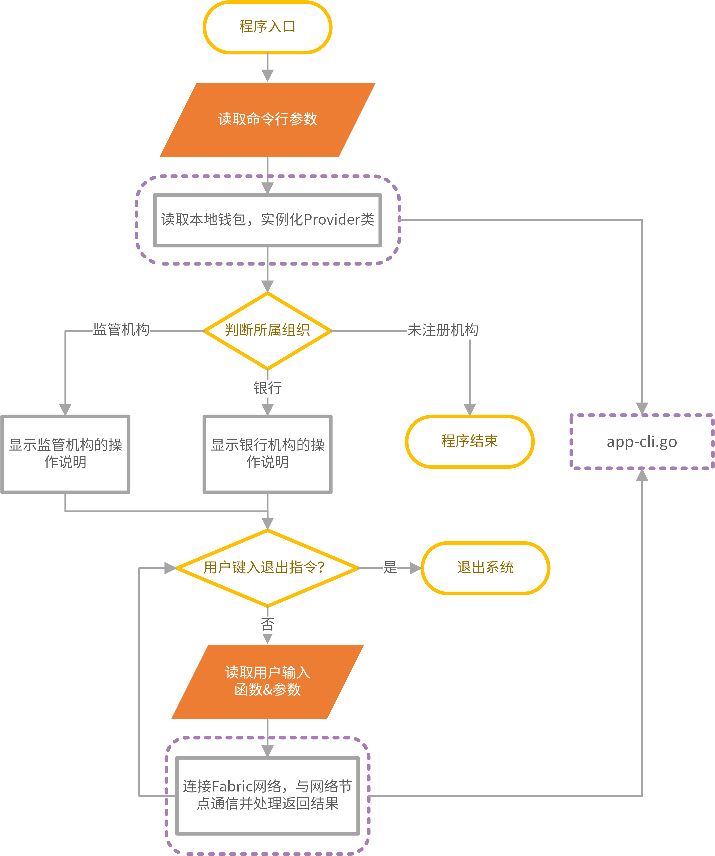
# Cli应用设计说明

本应用的设计用途为：为普通用户提供一个最为基本的操作界面，简化与区块链网络各个组件的交互流程，并提供最基本的操作说明和错误处理流程，可以视为银行端UI的原型。

# 程序结构

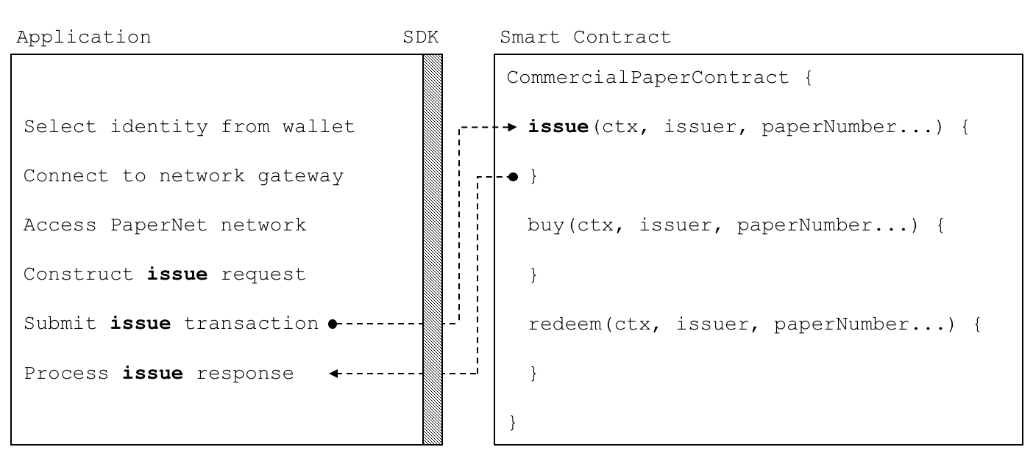
本程序的主体部分分布在 “/gopenbanking-cli.go” 和 “/app/cli-lib.go” 两个源文件中，前者为用户提供交互界面，处理用户输入和消息返回；后者为应用与区块链网络的交互提供支持和封装。两者的程序结构如图所示：

|  |  |
| --- | --- |
|  |  |
| /app/cli-lib.go | /gopenbanking-cli.go |

程序交互的流程如右图所示，交互界面部分的逻辑较为简单直接，可以用该图全面概括；区块链交互的部分则全部封装在 app-cli.go 中，并向外暴露New() 和 Provider.Invoke() 两个接口，分别实现虚线所示的客户端类初始化、区块链网络交互两个子流程。接下来的大部分内容为对这两个接口的设计进行解释。

## cli-lib.go

该文件隶属于 “app” 包，主要依赖于Hyperledger Fabric SDK所提供的接口，在本地钱包（即为/crypto-config/目录）及网关配置文件（即为/app/config.yaml）配置有效且固定的前提下，实现本地用户鉴权及智能合约的调用。



Fabric-SDK的典型调用流程

上图是应用程序调用网络中的智能合约（链码）的典型交互流程，但和常规的本地函数调用不同的是，调用链码是以 “transaction proposal” 的形式提交进区块链网络中的。所以在应用开发的过程中，我们不需要关注网络内部的具体配置，只需要确定用户身份有效且具有所需权限、生成 transaction proposal 即可。

### New()

该接口的主要作用是生成Provider类，其存储了用户的身份信息；设置环境变量以在接下来读取 config.yaml 的过程中使用；填充组织名->本地钱包目录的字典以便定位用户的MSP (Membership Service Provider) 信息；并实例化SDK调用所需的fabsdk实例，读取 config.yaml，及调用Provider.identify() 函数进行身份确权。

简而言之，以上操作的主要目的是为了与SDK的设计相衔接。由于SDK本身设计的一些问题（及bug），部分操作被复杂化了——如domainMap和环境变量的使用，但这已经是我们踩坑之后的最稳妥选择。更加具体的说明，请见代码注释。

### Provider.Invoke()

该接口负责与SDK的Query() 或 Execute() 两个主要功能交互。它本身的流程非常简洁：生成channel client 实例，并将用户输入的函数和参数进行格式转换生成 Request 实例；将请求传入网络并处理返回值。详细说明见代码注释。

# 配置文件

为了与SDK框架交互，在程序之外我们还需要另外配置网关配置文件 config.yaml（名字可自定）。该文件主要指定了Orderer、Peer等网络组件的地址和通信方式，以及需要指定本地钱包的地址。前者需要与网络管理员沟通，与服务器端的配置同步；后者可以通过传入环境变量的形式，避免需要写入绝对路径的情况（否则每个组织、每个用户都需要一份单独的文件）。

配置文件部分较为冗杂，可以参考本项目的配置，或Fabric-SDK-go 项目内提供的样例，参考文档注释进行尝试。