# 控制中心后端开发插件文档

这里以名为Demo的插件来进行说明。

## 插件代码

* 新建demo-plugin.h文件，定义插件类并继承Plugin：

#include “plugin\_i.h”

class PluginDemo : public Kiran::Plugin

{

public:

PluginDemo();

virtual ~PluginDemo();

virtual void activate();

virtual void deactivate();

};

主程序在激活插件时会调用activate函数，在卸载插件时会调用deactivate函数，因此插件的初始化逻辑可以放在activate函数中，析构/资源释放等逻辑可以放在deactivate函数中。

* 新建demo-plugin.cpp文件，添加接口宏定义：

PLUGIN\_EXPORT\_FUNC\_DEF(DemoPlugin)

该宏定义了两个接口函数：创建插件对象和析构插件对象，宏展开后的代码：

extern “C” void \*new\_plugin()

{

return new Kiran::DemoPlugin();

}

extern “C” void \*delete\_plugin(void \*plugin)

{

delete (Kiran::DemoPlugin\*)plugin;

}

这两个函数只会在主程序初始化和结束时调用一次，所以如果只能做一次初始化和一次析构操作的逻辑可以放在DemoPlugin的构造函数和析构函数中实现(一般不会用到)，其它初始化和析构逻辑还是放在activate/deactivate函数中实现。

## **插件安装**

### 插件安装目录

如果是system插件，则需要将so安装到/usr/lib64/kiran-cc-daemon/system/目录中；如果是session插件，则需要安装到/usr/lib64/kiran-cc-daemon/session/目录中。

### 插件配置

* 如果是和kiran-cc-daemon项目存放在一起，则需要修改data/plugin\_options.session或者data/plugin\_options.system文件，添加自己的插件配置，还是以前面的插件名Demo为例：

[demo]

Name=Demo

Description=Demo plugin

Available=true

group name的名字需要与so名字对应，so和group name的对应关系为：

so文件名 = lib + ${group\_name} + .so

因此这里查找的so文件名为libdemo.so

Name为插件名，Description为插件描述，这两个字段暂时未使用，但还是建议保持规范填写。

Available字段表示插件是否可用，如果为true则会在主程序启动时创建并激活插件，否则不会创建插件。

* 暂不支持插件单独放入一个项目中，后续可以考虑在插件安装目录中添加一个plugion\_options.d文件夹，可以将自己的插件配置放到这个文件夹中。文件名需和group name名保持一致

## **通用库**

插件共用的功能放在lib文件夹中，需要强制使用的功能：

* 日志类(lib/base/log.h)
* dbus返回的错误码定义(lib/base/error.h)