



SUNGARD 金仕达

KSFT_ API
特别说明

■ 文档标识

文档名称	KSFT_API 特别说明
文档编号	KS/IRDG-KSFT-05-2014
版本号	<V3.1.0>
状况	<input type="radio"/> 草案 <input type="radio"/> 评审过的 <input type="radio"/> 更新过的 <input checked="" type="radio"/> 定为基线的

■ 修订历史

版本	日期	描述	修订者
V3.1.0	2014-6-4	创建文档，主要内容包括：1. API 环境编译说明；2. Mac App 开发指导；3. 常见编程工具使用方法等。	高 嵩

■ 正式核准

姓名	签字	日期

■ 分发控制

副本	接受人	机构

目录

Mac App 开发指导	3
1. KSFT_API 编译环境说明	3
2. App 开发指导	3
2.1. 加载头文件	3
2.2. 加载动态库	3
2.3. 设置 rpath	4
2.4. 加载授权文件	5
2.5. 其他注意事项	6
3. Tips	6

Mac App 开发指导

本章节介绍如何使用 KSFT_API 开发 Mac App。如无特别说明，后续所提到的 Xcode 版本均为 5.0.2

1. KSFT_API 编译环境说明

- 操作系统：Mac OS X 10.9.3
- 编译器：clang 500.2.79
- C++标准库：libc++
- 优化级别：O2

2. App 开发指导

2.1. 加载头文件

将 KSFT_API 头文件添加到工程中。

2.2. 加载动态库

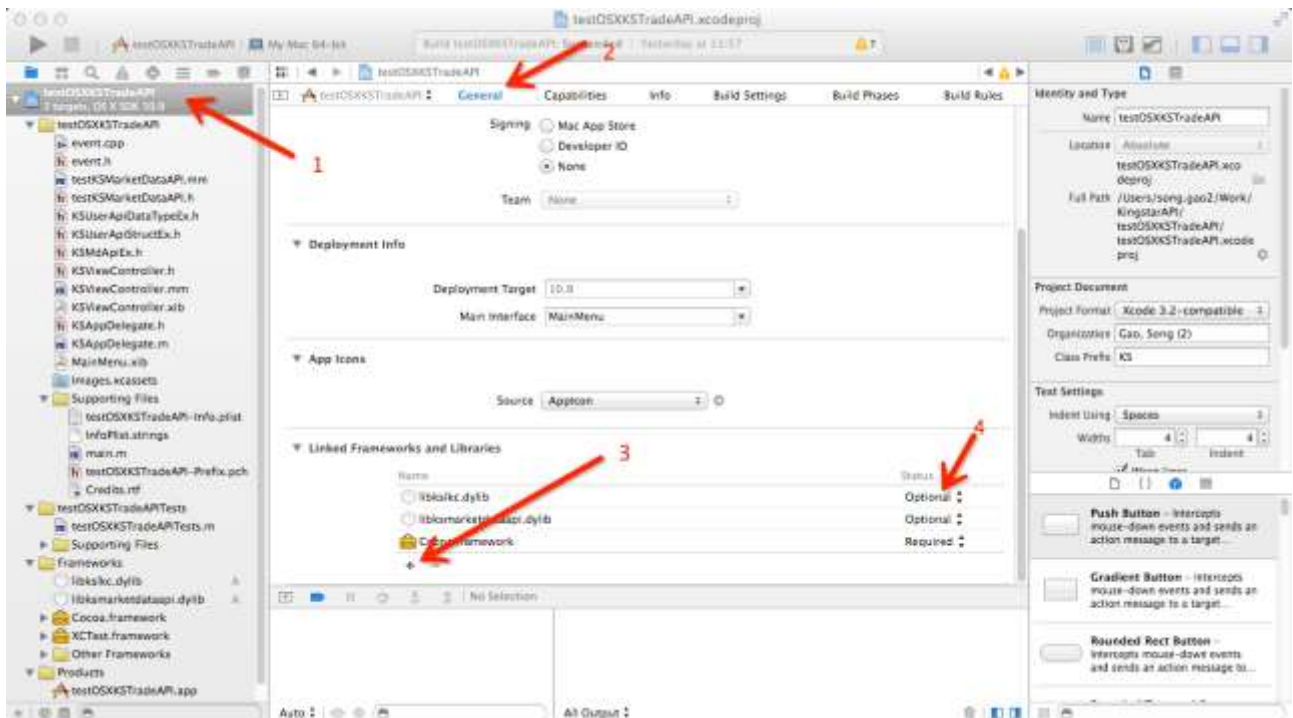


图 1 链接 KSFT_API C++动态链接库

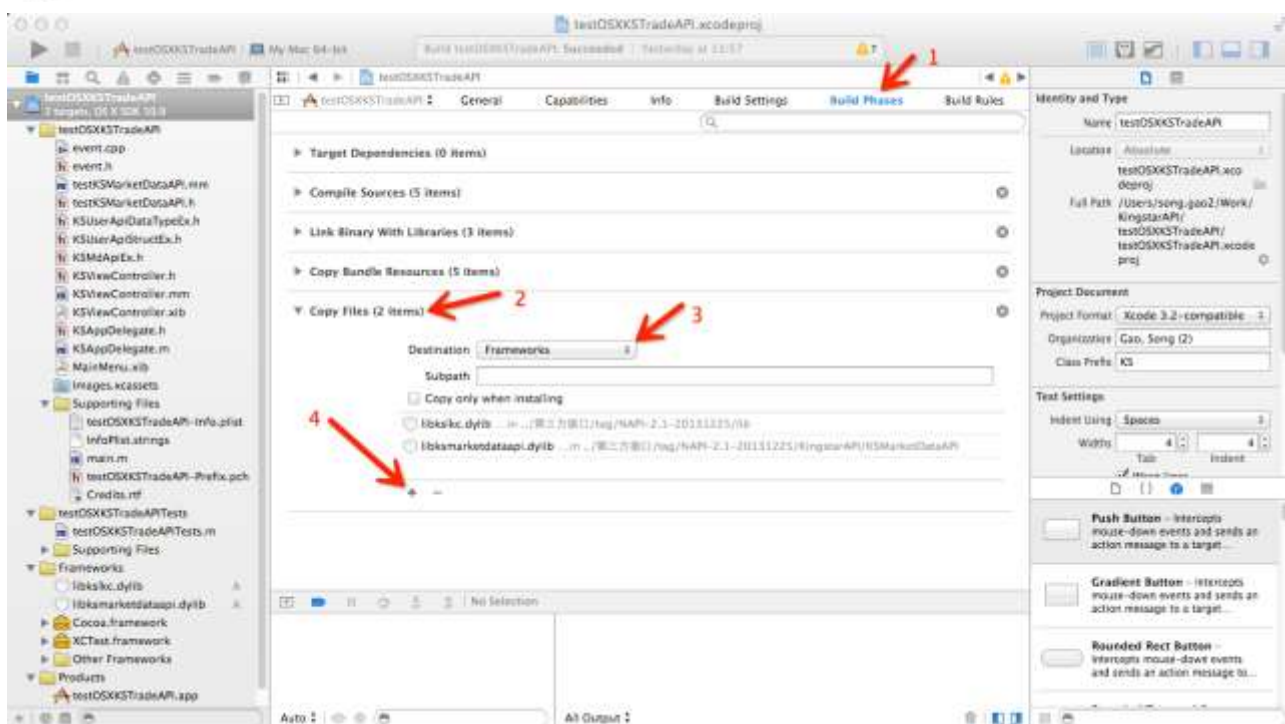


图 2 将动态链接库复制到 Bundle 中

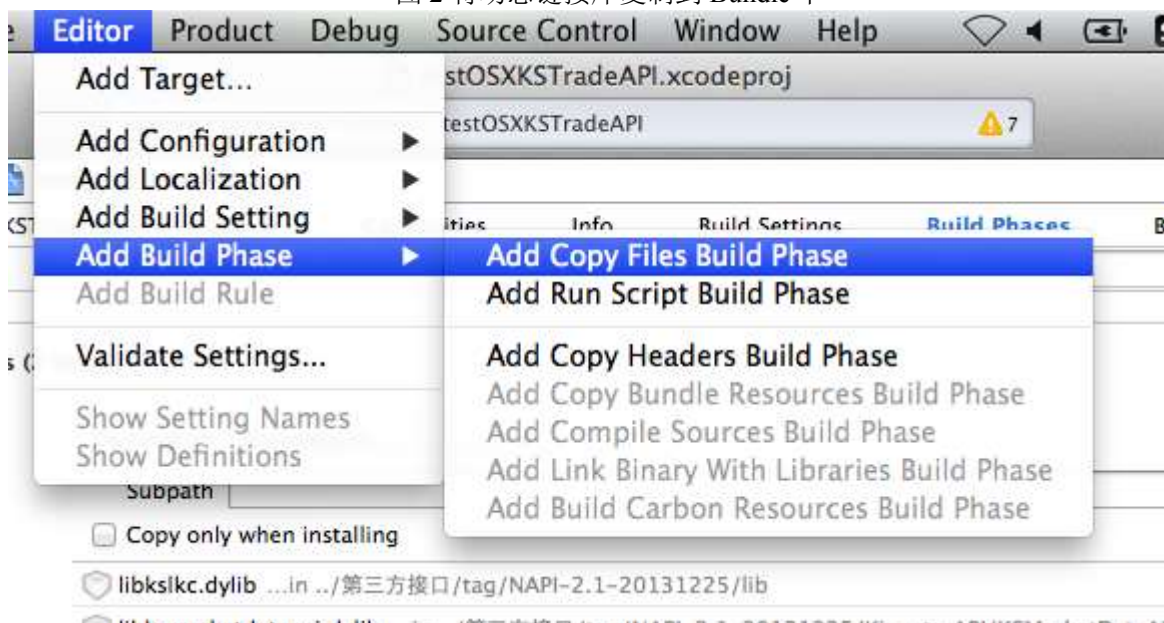


图 3 添加“Copy Files” Build Phase

说明:

1. 在 Xcode 中选中工程文件并选择“General”选项卡(如图 1, 标签 1、2);
2. 在“Linked frameworks and libraries”组添加 KSFT_API C++动态链接库至该工程并将动态链接库右侧的 Status 改为“Optional”(如图 1, 标签 3、4);
3. 切换到“Building Phases”选项卡(如图 2, 标签 1)并添加“Copy Files”Phase, 如图 3;
4. 将 Destination 更改为“Frameworks”, 并点击“+”按钮把 API 动态链接库添加其中(如图 2, 标签 3、4);

2.3. 设置 rpath

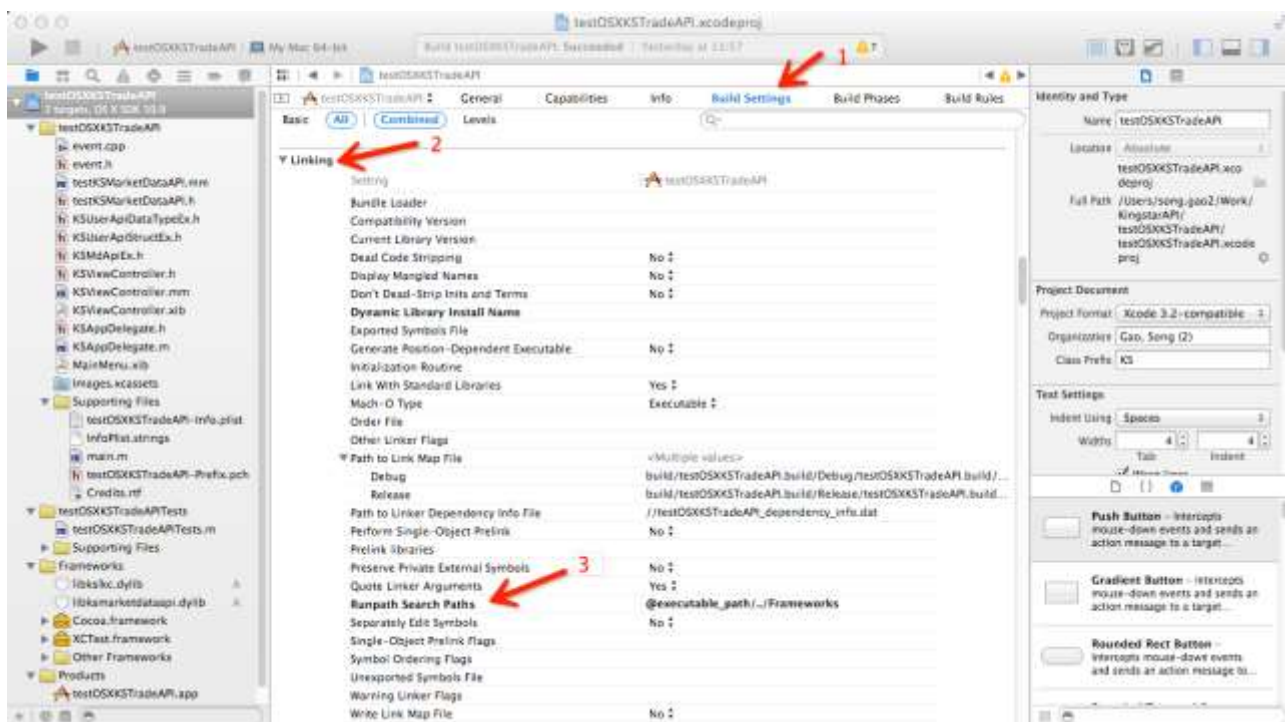


图 4 设置 rpath

说明:

1. 切换到“Build Setting”选项卡并找到 Linking 组（如图 4，标签 1、2）；
2. 将 Runpath Search Paths 设置成“@executable_path/../Frameworks”（如图 4，标签 3）；

2.4. 加载授权文件

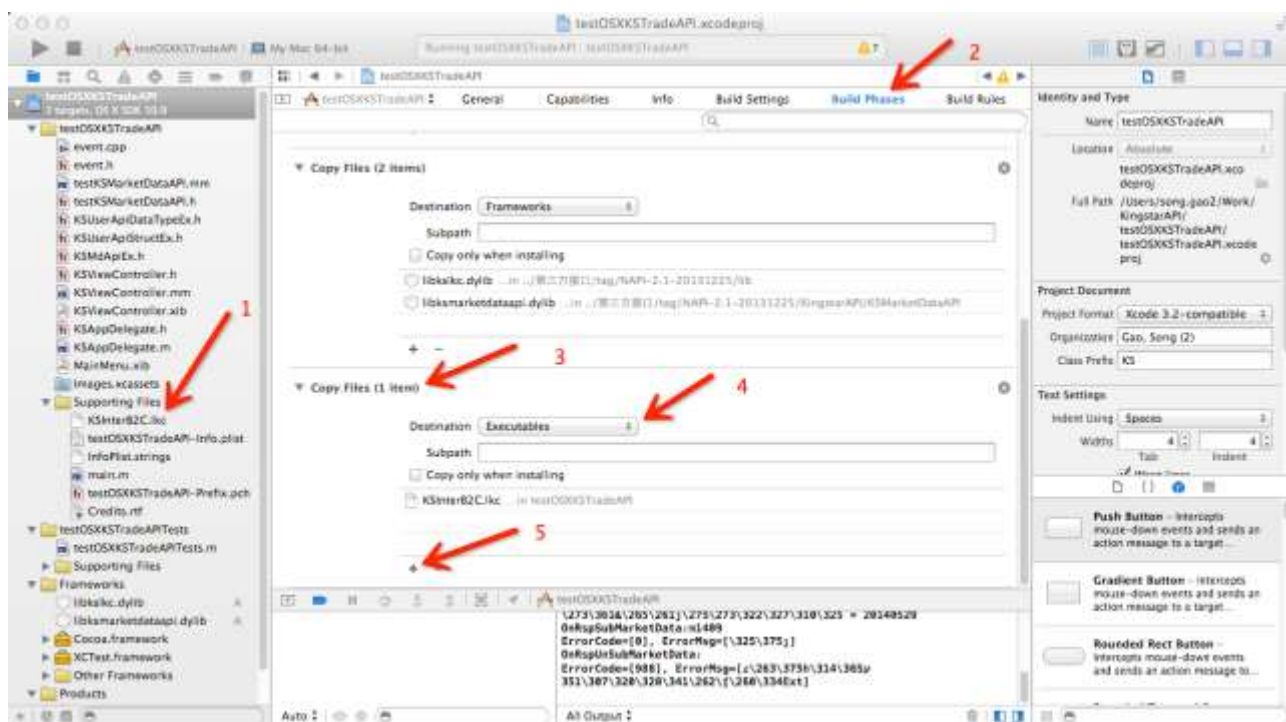


图 5 加载授权文件

说明:

1. 将授权文件“KSInterB2C.lkc”添加到工程中（如图 5，标签 1）；
2. 切换到 Build Phases 选项卡并添加“Copy Files”Phase，如图 3；
3. 将 Destination 设置为“Executable”，并点击“+”，将授权文件添加其中（如图 5，标签 4、5）；

2.5. 其他注意事项

KSFT_API 采用 C++ 编写，需要与 Objective-C++ 共同协作，故请将 Objective-C++ 源代码文件的后缀改成 mm。否则，会导致编译或链接错误。

3. Tips

1. 查看动态链接库或可执行文件依赖的程序集

`$ otool -L <dylib or executable>`

示例：

```
AP-CHN-LP140007:KSTradeAPI song.gao2$ otool -L libkslkc.dylib
libkslkc.dylib:
    @rpath/libkslkc.dylib (compatibility version 0.0.0, current version 0.0.0)
    /usr/lib/libc++.1.dylib (compatibility version 1.0.0, current version 120.0.0)
    /usr/lib/libSystem.B.dylib (compatibility version 1.0.0, current version 1197.1.1)
```

2. 查看动态链接库的 install name

`$ otool -D <dylib>`

示例：

```
AP-CHN-LP140007:KSTradeAPI song.gao2$ otool -D libkstradeapi.dylib
libkstradeapi.dylib:
    @rpath/libkstradeapi.dylib
```

3. 更改动态链接库的 install name

`$ install_name_tool -id <install name> <dylib>`

示例：

```
AP-CHN-LP140007:KSTradeAPI song.gao2$ install_name_tool -id libkstradeapi.dylib libkstradeapi.dylib
AP-CHN-LP140007:KSTradeAPI song.gao2$ otool -D libkstradeapi.dylib
libkstradeapi.dylib:
    libkstradeapi.dylib
```

4. 查看库所支持的 CPU 架构

`$ lipo -info <library>`

示例：

```
AP-CHN-LP140007:KSTradeAPI song.gao2$ lipo -info libkstradeapi.dylib
Architectures in the fat file: libkstradeapi.dylib are: i386 x86_64
```

5. 从通用动态链接库中提取特定 CPU 架构的库

`$ lipo -extract <arch_type> -o <output> <universal dylib>`

示例：

```
AP-CHN-LP140007:KSTradeAPI song.gao2$ lipo -info libkstradeapi.dylib
Architectures in the fat file: libkstradeapi.dylib are: i386 x86_64
AP-CHN-LP140007:KSTradeAPI song.gao2$ lipo -extract i386 -o libkstradeapi_i386.dylib libkstradeapi.dylib

AP-CHN-LP140007:KSTradeAPI song.gao2$ ls
KSTradeAPI.h          libkslkc.dylib          libkstradeapi.dylib      libkstradeapi_i386.dylib
AP-CHN-LP140007:KSTradeAPI song.gao2$ lipo -info libkstradeapi.dylib
Architectures in the fat file: libkstradeapi.dylib are: i386 x86_64
AP-CHN-LP140007:KSTradeAPI song.gao2$ lipo -info libkstradeapi_i386.dylib
Architectures in the fat file: libkstradeapi_i386.dylib are: i386
```

6. 制作 Universal 动态链接库

\$ lipo -create <i386 dylib> <x86_64 dylib> -o <dylib>

示例:

```
AP-CHN-LP140007:KS_API song.gao2$ ls
libksmarketdataapi_i386.dylib  libkstradeapi_i386.dylib
libksmarketdataapi_x86_64.dylib libkstradeapi_x86_64.dylib
AP-CHN-LP140007:KS_API song.gao2$ lipo -create libkstradeapi_i386.dylib libkstradeapi_x86_64.dylib -o libkstradeapi.dylib
AP-CHN-LP140007:KS_API song.gao2$ lipo -info libkstradeapi_i386.dylib
Non-fat file: libkstradeapi_i386.dylib is architecture: i386
AP-CHN-LP140007:KS_API song.gao2$ lipo -info libkstradeapi_x86_64.dylib
Non-fat file: libkstradeapi_x86_64.dylib is architecture: x86_64
AP-CHN-LP140007:KS_API song.gao2$ lipo -info libkstradeapi.dylib
Architectures in the fat file: libkstradeapi.dylib are: i386 x86_64
```

7. 编译时给链接器传递 rpath

\$ clang++ -Wl,-rpath -Wl,<run path> ...

示例参见测试程序的 makefile。