# iOS 10 SiriKit QQ 适配详解

0

iOS (http://www.csdn.net/tag/iOS/news) Siri (http://www.csdn.net/tag/Siri/news)
SiriKit (http://www.csdn.net/tag/SiriKit/news)
QQ (http://www.csdn.net/tag/QQ/news)
适配 (http://www.csdn.net/tag/适配/news)

阅读 3592



**声明：** 本文为腾讯Bugly开发者社区 (https://bugly.qq.com/)投稿，作者：罗鑫，非经作者同意，请勿转载。
原文地址：http://dev.qq.com/topic/57ece0331288fb4d31137da6 (http://dev.qq.com/topic/57ece0331288fb4d31137da6)

## 1. 概述

苹果在 iOS 10 开放了 SiriKit 接口给第三方应用。目前，QQ 已经率先适配了 Siri 的发消息和打电话功能。这意味着在 iOS 10 中你可以直接告诉 Siri 让它帮你发 QQ 消息和打 QQ 电话了，听起来是不是很酷炫？

那么第三方应用使用 Siri 的体验究竟如何？哪些应用可以接入 SiriKit？接入 SiriKit 又需要做哪些工作呢？这篇文章会为你一一解答这些疑惑。


*图1 用 Siri 发 QQ 消息效果展示*

# 2. SiriKit 简介

我们都知道 Siri 是 iPhone 手机中的智能语音助手，那么什么是 SiriKit 呢？SiriKit 是苹果为第三方应用支持 Siri 提供的开发框架。在官方文档中，SiriKit 将对不同场景的语音支持划分为不同的 domain，目前，SiriKit 支持的 domain 包括：VoIP 电话、发消息、转账、图片搜索、网约车订车、CarPlay 和餐厅预定，也就是说如果你的应用中包含有这些功能之一，就可以考虑将这些功能接入到 SiriKit 中啦。

实现 SiriKit 相关功能时，我们并不需要真正对语音进行识别，语音的识别工作会由 Siri 完成。Siri 识别完语音后，会将语音要完成的功能抽象成 Intent 对象传递给我们，而我们的接入工作主要是与这些 Intent 对象打交道，并不会涉及到自然语言处理（NLP）的技术。

关于 SiriKit 的开发网上已有一些文章（比如《SiriKit 初探 —— WWDC 2016 技术赏析 (http://geek.csdn.net/news/detail/100195)》），也可参考苹果的官方文档 SiriKit Programming Guide (http://developer.apple.com/library/prerelease/content/documentation/Intents/Conceptual/SiriIntegrationGuide/index.html)，本文着重介绍 QQ 的适配经验。

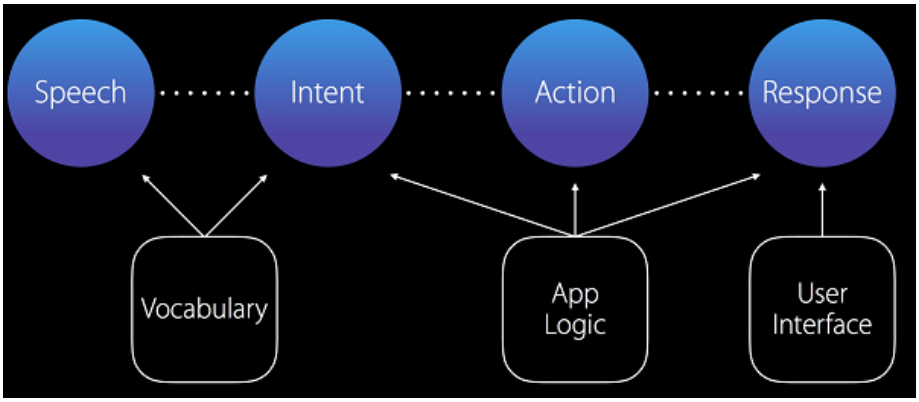


*图2 SiriKit 原理*

# 3. SiriKit 接入

要实现 SiriKit 的功能需要在 Xcode 工程中添加 Intents Extension 的 target，和其他 extension 一样，Intents Extension 是一个独立于 Containing App 进程运行的插件，主要用于处理和确认来自 Siri 的 intent 请求。如果想让 Siri 在处理 App 相关 intent 时提供一些自定义的界面，那么你就需要再添加 Intents UI Extension 的 target，Intents UI Extension 也是一个独立运行的插件（所以要完整的支持 SiriKit 其实是需要添加两个 target，有点蛋疼）。关于 App Extension 的开发可以参考苹果的 App Extension Programming Guide (https://developer.apple.com/library/content/documentation/General/Conceptual/ExtensibilityPG/ExtensionCreation.html)。

**我们以 QQ 中的发消息功能为例说明一下 SiriKit 的接入方法：**

首先，我们需要在 Intents Extentsion 的 info.plist 文件中配置须支持的 Siri Intents，在 IntentsSupported 中加入 INSendMessageIntent，如果需要在锁屏时禁用某个功能，则再在 IntentsRestrictedWhileLocked 中加入相应项的 Intent，如图3所示。

| | | |
|---|---|---|
| ▼NSExtension | Dictionary | (3 items) |
| ▼NSExtensionAttributes | Dictionary | (2 items) |
| ▼IntentsRestrictedWhileLocked | Array | (2 items) |
| Item 0 | String | INStartVideoCallIntent |
| Item 1 | String | INStartAudioCallIntent |
| ▼IntentsSupported | Array | (3 items) |
| Item 0 | String | INSendMessageIntent |
| Item 1 | String | INStartAudioCallIntent |
| Item 2 | String | INStartVideoCallIntent |

*图3 Intent Extentsion info.plist 配置*

SiriKit 的接入主要分为 Intents Extension 和 Intents UI Extension 两部分，下面分别进行介绍。

## Intents Extension

当我们对 Siri 说“用 QQ 发消息给王一然说你好”时，语音的识别将会由 Siri 自动完成，Siri 会将识别好的内容展示在 Siri 的界面。如图4所示，我们可以看到一个完整的发消息语句主要由四部分组成：

1. **应用名**：告诉 Siri 要使用哪个 App，Siri 会根据 App 的 bundle displayname 自动识别 App 的名称，无需额外注册。
2. **发消息 Intent**：告诉 Siri 要使用发消息的功能，我们实测发现说发信息也是能识别，具体还有哪些词汇会识别为发消息的 intent 苹果没有在文档中说明。
3. **消息接收者**：告诉 Siri 消息的接收者是谁，“王一然”是我 QQ 好友的昵称。
4. **消息内容**：告诉 Siri 你要发的消息内容是什么，这里的消息内容为“我很生气”。


*图4 确认发送消息界面*

其中应用名和 Intent 是必须的，不然 Siri 无法抽象出你的“Intent”。后两项如果缺省的话，我们可以在实现中要求用户进一步提供数据或者忽略。在识别完成后 Siri 会将消息内容和接收者抽象成一个 INSendMessageIntent 传递给 QQ 的 Intent Extension。

我们从图4还可以看到 Siri 准确从我的语音中识别出我 QQ 好友中昵称为“王一然”的好友，然而“王一然”并不是一个通用的短语，那么这是怎么做到的呢？奥秘就在于在 QQ 运行时我们把所有 QQ 好友的昵称同步到了 Siri 云端，这样 Siri 就可以识别出特定用户要使用的特定短语，详细同步方法可参考 INVocabulary 的 `setVocabularyStrings:ofType:` 方法。

每个 domain 的功能在 Siri 中都有对应的 Intents，而每个 intents 都对应一个特定的 handler 协议。对于发消息来讲，对应的 Intent 和 handler 协议分别为 `INSendMessageIntent` 和 `INSendMessageIntentHandling`。只要实现 `INSendMessageIntentHandling` 协议中的相关方法，并在 Siri 解析出 `INSendMessageIntent` 请求时用我们的 `INSendMessageIntentHandling` 对象去处理相关的发消息请求。具体的流程如图5：

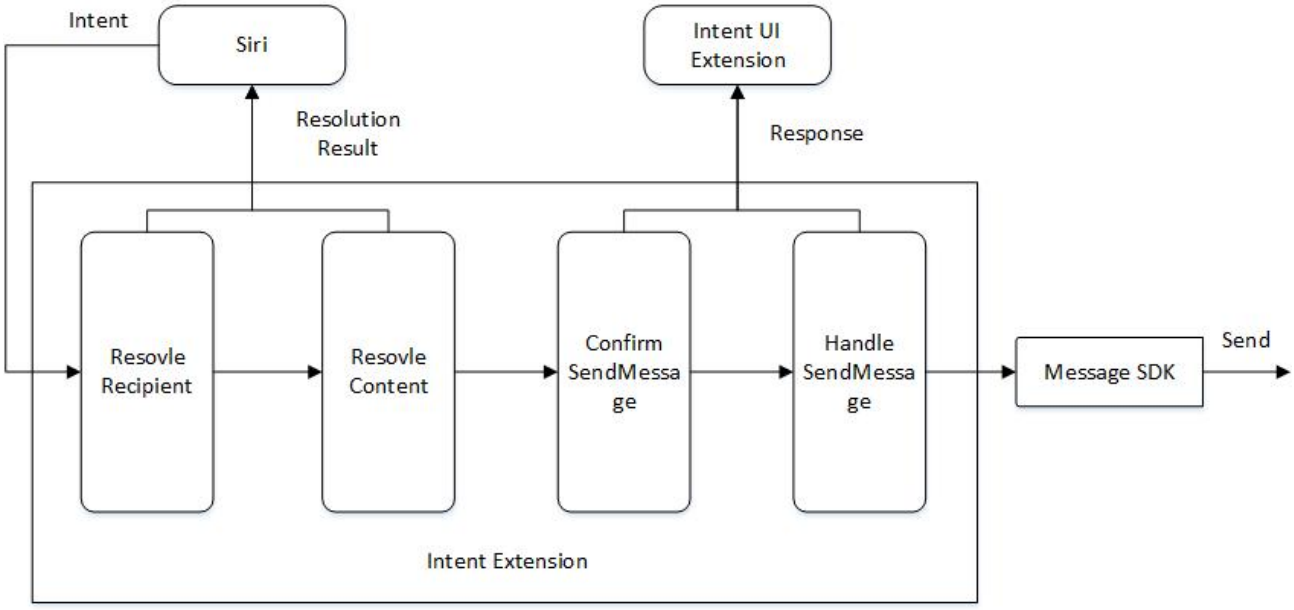


*图5 Siri 发 QQ 消息流程*

### 1）ResolveRecipientsForSendMessage

对 Siri 从Intent中传递过来的接收者名称进行处理和确认，比如可以确认该名称当前是否在 QQ 好友列表中，并将 resolution result 反馈给 Siri。Resolution result 代表了应用对 intent 处理后的结果，对于发消息来说，表1列举了几种可能的 resolution results。

*表1 send resolution result*

| Resolution results | 说明 |
| --- | --- |
| Success | 接收者确认成功，符合要求 |
| Need value | 无接收者，需要用户继续确认接收者 |
| Not require | 表明忽略这个处理，即使没有接收者也可以继续 |
| Disambiguation | 存在多个相同的接收者，需要用户选择 |
| Unsupported. | 不支持的接收者 |



**2）ResolveContent**

与接收者的处理类似，在这个方法中可以对 Siri 识别出的消息内容进行“修饰”，并且将 resolution result 反馈给 Siri，比如 QQ 对一些消息里面的特殊词汇如“生气”做了 emoji 适配。

**3）ConfirmSendMessage**

这个方法的作用是确认是否要发送该消息，可以在这一步进行一些鉴权工作，鉴权通过后再确认发送，否则取消。确认可以发送后会调起确认发送界面，如图4所示。如果需要从Containing App共享数据，具体的实现方案参考 App Group 的 Shared Container。

4）HandleSendMessage

如图4，当用户点击了“发送”按钮或者用语音给出了发送指令时会最终进入到这个方法，在这个方法里我们需要实现发消息的逻辑，发送成功后可以调起消息发送成功的界面，如图6。


*图6 消息发送成功界面*

## Intents UI Extension

对于支持自定义界面的 Intent 类型，可以在 Intents UI Extension 中提供更美观的自定义界面。 Custom UI 的实现相对较简单，和 iOS App 的开发一样，都是通过 UIViewController 的子类实现。我们需要在 Intents UI Extension 的 info.plist 文件中设置 initial viewcontroller 或者设置 main storyboard，对于不同类型的 Intent 的界面展示，通过 Child Viewcontrollers 的方式实现差异化界面展示。

如图7所示，当接收到来自 Intents Extension 的 response 时，系统会唤起 Intents UI Extension 并加载 initial viewcontroller，通过 `INUIHostedViewSiriProviding` 协议的 `configureWithInteraction:context:completion:` 方法可以获取 intent，比如在发消息功能中，在消息确认发送和发送成功后都会回调一次这个方法。根据 Intent 对象的类型和状态，在收到相关 Intent 的回调时 present 对应的 Child Viewcontroller 即可实现定制化的界面展示。

这里需要注意的是，Intents UI Extension 的进程并不会在界面销毁后就退出，很可能只是在后台处于休眠状态，下次 response 到来时再被唤醒。

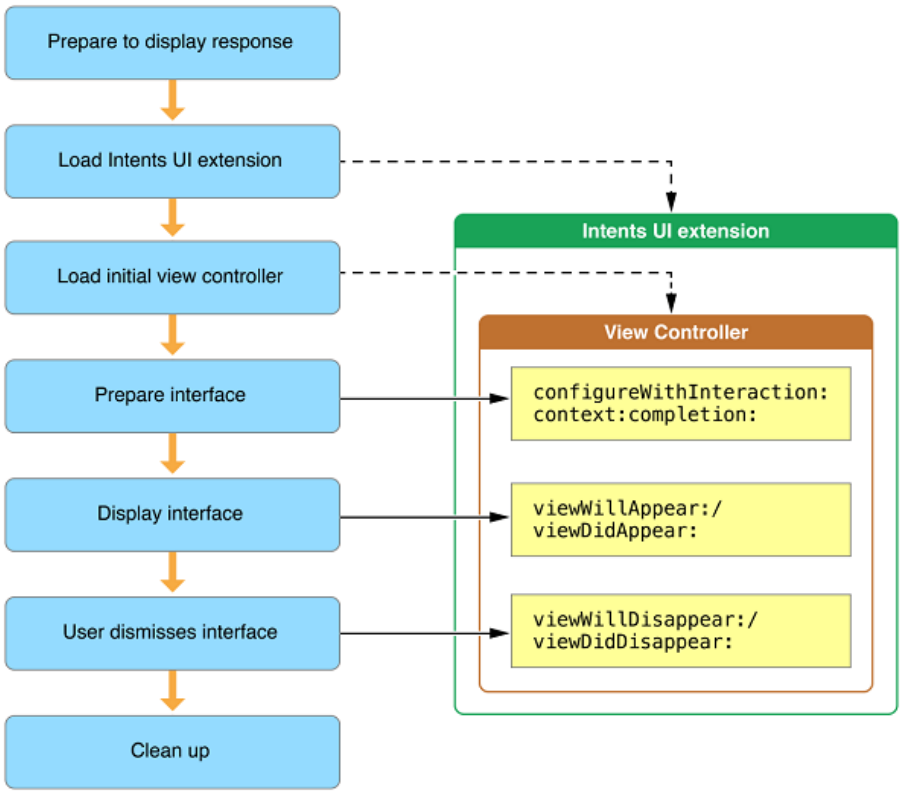


*图7 Life cycle of an Intents UI extension*

# 4. 总结

总的来说虽然苹果这一次对 SiriKit 开放的场景有限，但是从我们的适配经历来看苹果对 Siri 还是非常重视的。另外，这是 SiriKit 首次对第三方应用开放接口，所以不可避免存在一些问题。我们在开发过程中也确实遇到了一些 SiriKit 本身的 Bug，大部分 Bug 在向苹果反馈后都得到了解决，但是在语言识别方面 Siri 依然存在一些缺陷，比如对中英文混合的场景识别依旧不太好。期待以后 Siri 对中文的支持越来越好，也希望 Siri 能够开放更多的场景给第三方应用适配。

---



(http://geek.csdn.net/user/publishlist/byeweiyang)
CSDN移动 (http://geek.csdn.net/user/publishlist/byeweiyang)
发布于 iOS开发 (http://geek.csdn.net/forum/66) 2016-10-11 13:41

已有3条评论 **最新**

评论

0
hankeet (http://geek.csdn.net/user/publishlist/hankeet) 20小时前
(http://geek.csdn.net/user/publishlist/hankeet)
值得学习一下

0
hisourcezhang (http://geek.csdn.net/user/publishlist/hisourcezhang) 2016-10-11 15:46
(http://geek.csdn.net/user/publishlist/hisourcezhang)
mark,新东西以后可能会用！

0
Chay Tang (http://geek.csdn.net/user/publishlist/tangchengyou) 2016-10-11 14:30
(http://geek.csdn.net/user/publishlist/tangchengyou)
记录一下。