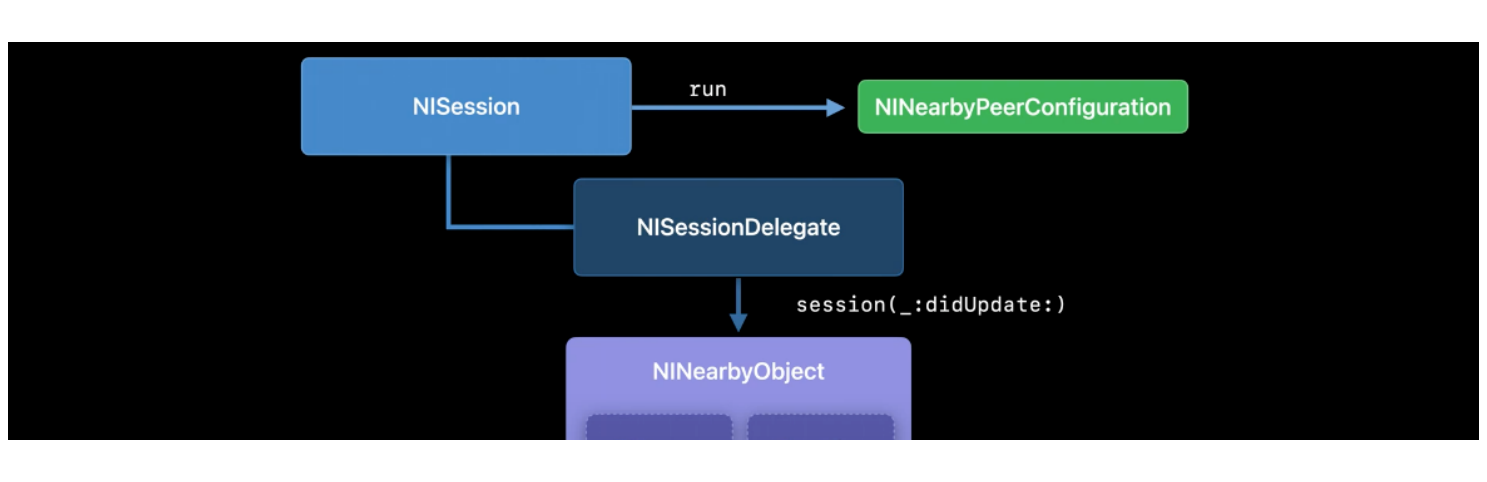


【WWDC21 10165】探索与第三方硬件的近距离交互

老司机技术周报 + 关注
预计阅读时间12分钟 7 月前

> 作者：大猫，iOS开发
>
> 审核：
>
> LoneyIsError，中移（苏州）软件技术有限公司 研发工程师
>
> Damonwong，iOS 开发，老司机技术周报编辑，就职于淘系技术部

基于 Session 10165 探索与第三方硬件的近距离交互

2021 年 4 月的发布会上，AirTag 横空出世，从此妈妈再也不用担心我找不到钥匙了。UWB 这项技术也慢慢走进了人们的视野。现在不仅仅是只有官方的钥匙扣了。第三方的硬件授权设备也能与苹果进行交互了。
本文是讲了苹果在 Nearby Interaction 框架上的更新，全文主要分为三部分：
首先带你快速的了解一下之前是如何使用 Nearby Interaction 框架的，然后讲述在 iOS 15 上关于用户访问权限的改进，最后根据最新的 api 来实现一个简单的 demo。

快速回顾一下如何使用 Nearby Interaction框架

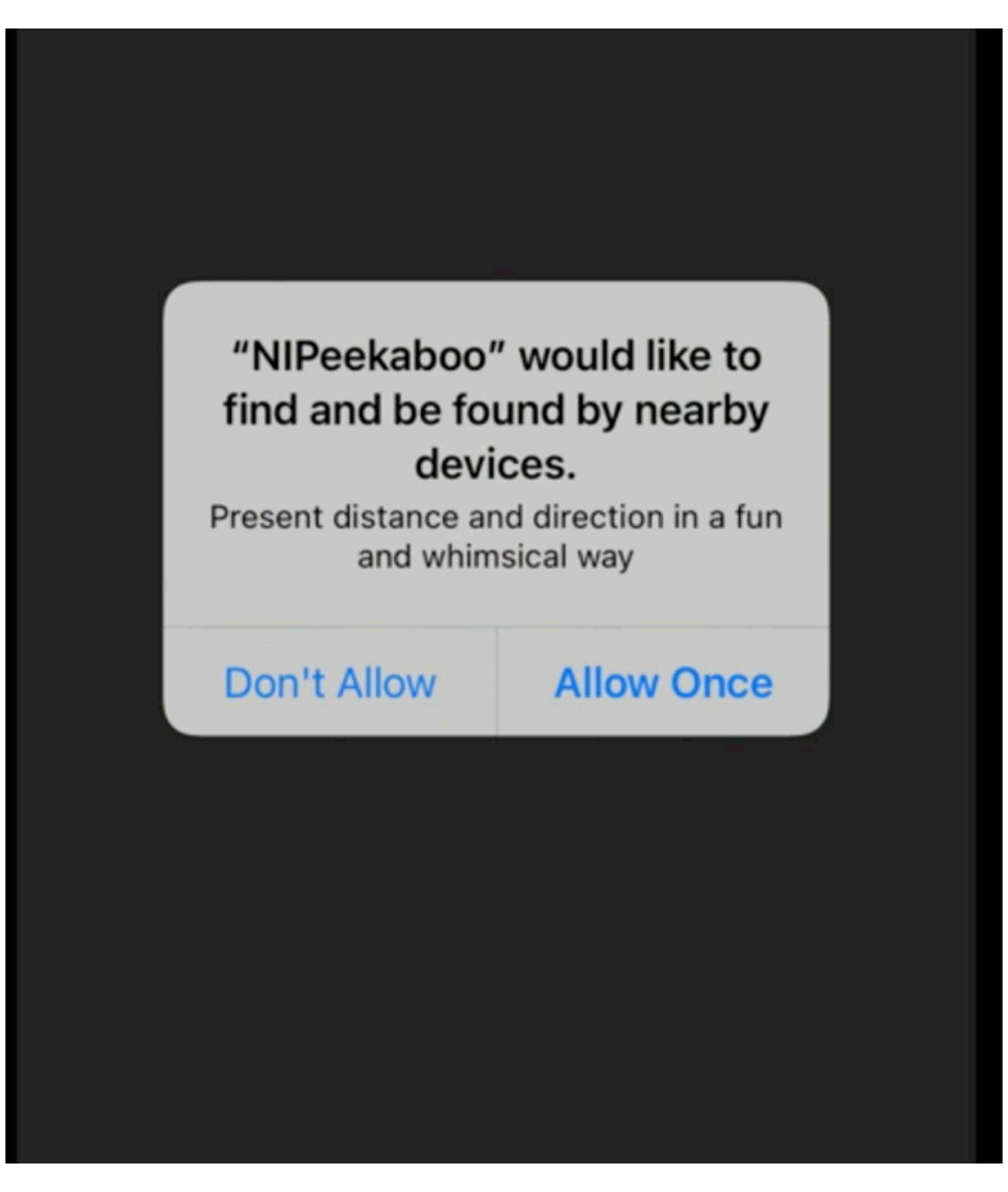
```
class ViewController: UIViewController, NISessionDelegate {

    override func viewDidLoad() {
        super.viewDidLoad()
        let token = _token
        let niSession = NISession()
        niSession.delegate = self
        niSession.run(NINearbyPeerConfiguration(peerToken: token))
    }
    func session(_ session: NISession, didUpdate nearbyObjects: [NINearbyObject]
){
        nearbyObjects.forEach { (niNearbyObject) in
            print(niNearbyObject.distance!);
            print(niNearbyObject.direction!);
        }
    }
}
```

如何在两个手机之间建立联系呢？首先创建一个 NISessions 实例，然后在代码中遵守 NISessionDelegate 协议，在 session 实例的 run 函数中需要传入一个 NIConfiguration 的子类做配置对象。如果想要在两个 iPhone 之间建立联系，可以使用 NINearbyPeerConfiguration 做参数，但是在 NINearbyPeerConfiguration 的初始化中需要知道对方的 token，这个 token 可以通过网络进行传输，当你准备好这一切的时候，Nearby Interaction 将开始为程序提供 NearbyObject 更新流，每个更新包含到附近设备的距离和方向（可选）。如果想要继续深入的了解这个库的 API，可以去看去年演讲 "Meet Nearby Interaction"。

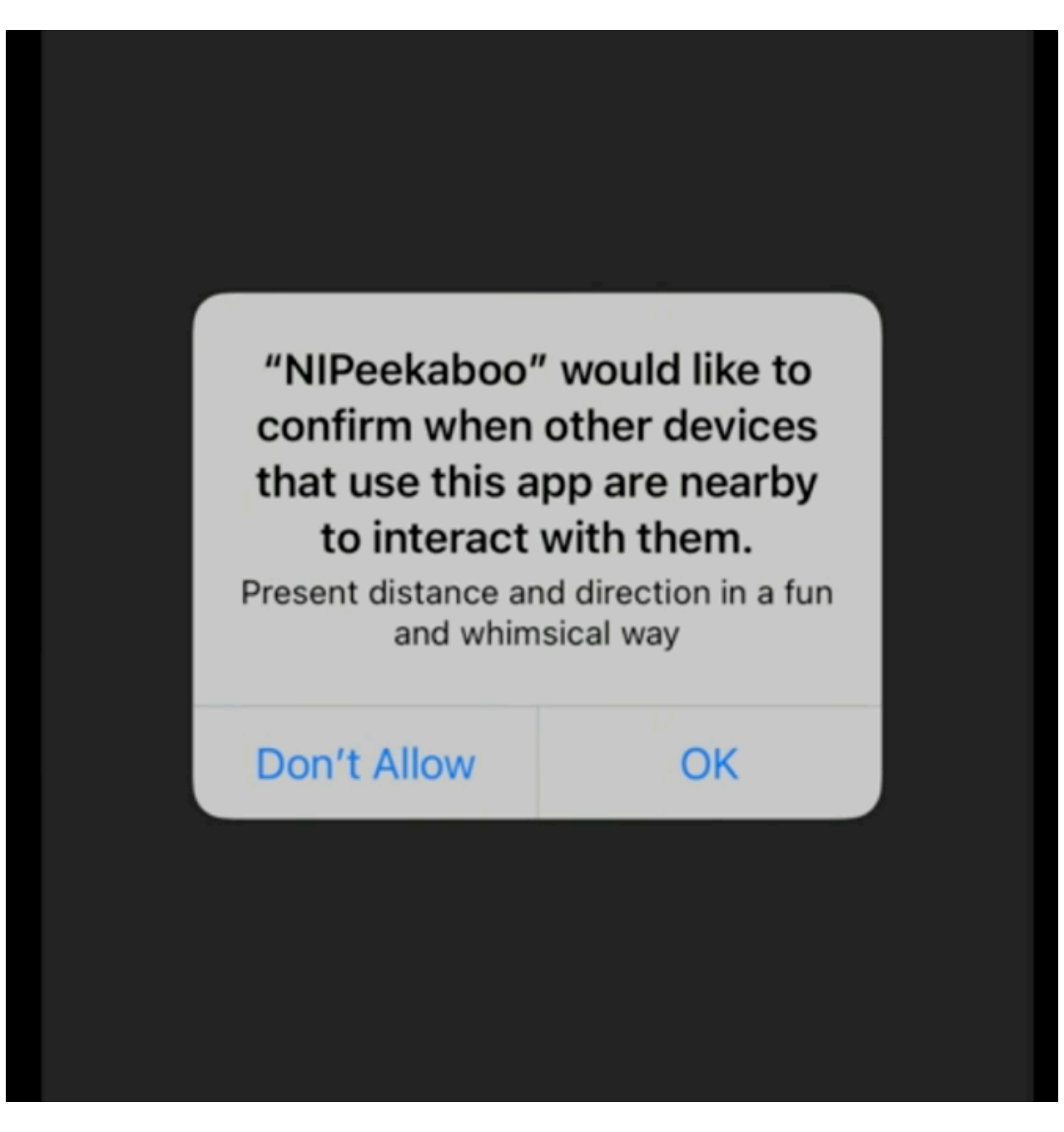
用户权限上的改进

在 iOS 14 上的的 Nearby Interaction 的权限流程是这样的，当你的 App 在一个新的生命周期中，第一次创建一个 NISession 会话的时候会展示弹窗，弹窗的样式如下图，但由于此权限是一次性的，因此在某些情况下也可能导致展示多次弹窗。



"NIPeekaboo" would like to find and be found by nearby devices.
Present distance and direction in a fun and whimsical way

今年在 iOS 15 上对权限进行了改进，弹窗如下图。这是 iOS 15 中新的“附近交互”权限提示。看起来很相似，但却和之前的权限授予流程有以下不同：



"NIPeekaboo" would like to confirm when other devices that use this app are nearby to interact with them.
Present distance and direction in a fun and whimsical way

- 提示时机：系统会在应用程序第一次运行 NISession 时自动显示权限提示。因此确保运行 NISession 的时间与用户想要使用这个功能的时间保持一致，这样您的用户就很容易理解为什么您需要这个权限。
- 提示的选项：让我们仔细看看提示上的新选项。新的“确定”选项可在应用程序使用时授予您的应用程序权限。无论用户接受还是拒绝您的应用使用附近交互的请求，权限提示都不会再次显示。
- 新的权限设置位置：从 iOS 15 开始，使用 Nearby Interaction 的应用将出现在“设置”中。因此，如果用户改变主意，他们可以转到“设置”应用并更改应用的“附近互动”访问权限。应确保在开发应用时针对权限的变更进行测试。

总结一下新的 Nearby Interaction 用户权限模型。在弹窗中点击 Allow 后，您的应用将获得在应用使用过程中使用 Nearby Interaction 的持久权限。弹窗上将显示一个使用说明字符串，在 Info.plist 中配置。在这个配置中要简洁明了的解释您在应用程序中访问附近的设备可以提供那些有趣的能力。在第一次也是最后一次出现提示后，您的应用的名称和图标将出现在“设置”中，这意味着用户可以随时更改您的应用的权限状态。当应用程序没有足够的权限使用 Nearby Interaction 时， NISessions 所有的功能都会失效。因此，如果应用程序中的关键功能依赖于对附近设备的访问，请务必向用户清楚地解释这一点，并在适当的时候引导他们使用“设置”去打开配置权限。

从一个简单的需求入手讲一下新的 API

没有业务场景的新特性都是瞎逼逼，下面将从一个实用的业务场景出发来介绍如何使用 Nearby Interaction API 与第三方配件一起工作。

> 想象一下，在你的设备周围定义了一个半径为 1.5 米的区域和另外一个半径为 3 米的更大的区域，当用户进入较大的区域的时候您希望启动功能 A，当用户进入较小的区域的时候你期望启动功能 B，如何使用 Nearby Interaction 来实现这一个功能?

part1 确保硬件设备与程序之间具备数据通道

Nearby Interaction 需要硬件与应用程序之间具有数据交换的能力。至于采用何种技术实现数据交换，这完全取决于硬件设备具有什么样的能力。假设设备已经支持蓝牙。因为您将能够利用现有的蓝牙功能来进行数据交换。如果硬件设备连接到了本地网络或互联网中，完全可以使用网络进行数据交换，在应用程序和配件之间可以互相传输数据是下一步需要执行的操作的必要条件。

part2 基于 token 创建 NearbyAccessoryConfiguration

上面的回顾中我们在了两部 iPhone 之间启动会话时创建了 NearbyPeerConfiguration。如果要开始与硬件的会话，需要创建 NearbyAccessoryConfiguration。这是 iOS 15 中新的 NIConfiguration 类型。如果要实例化一个 NearbyAccessoryConfiguration。需要为它提供一些配置数据，系统接受的一种特定的用来描述这个设备的数据格式。但是我们如何获得这个配置数据，这个特定的格式是什么？与 U1 兼容的硬件设备（认证过的供应商）将知道如何根据请求生成此配置数据。这意味着您在硬件设备本身上运行的代码需要生成此数据然后通过 part1 的数据通道将其发送到您的应用程序。代码如下：

```
    private func setupAccessory(_ configData: Data, name: String) {
        do {
            config = try NINearbyAccessoryConfiguration(data: configData)
        } catch let error {
            print("Bad config data from accessory \(name). Error: \(error)")
            return
        }
        // 保存token
        cacheToken(config!.accessoryDiscoveryToken,accessoryName:name)
    }
```

setupAccessory 是我在应用程序中编写工具方法。每当我从硬件设备获取到配置数据时，都应该调用此方法。现在，我可以使用收到的数据创建 NINearbyAccessoryConfiguration 。尽量在 do/catch 语句中创建这个配置。这样做的好处是如果数据无效的话，NIConfiguration 的 init 方法会抛出异常。如果配置对象创建成功的话，就说明从硬件传输过来的数据格式是正确的。创建配置的最终目的是使用它来运行会话。与硬件设备进行交互。

part3 与硬件设备进行交互
现在，我们已经准备好与这个硬件进行交互了。为了管理交互，需要创建一个 NISession 实例，并设置他的代理，在启动会话时，需要在 NISession 实例的 run 函数中传入上面创建好的 Configuration。就像 Nearby Interaction 需要来自硬件的配置数据一样，硬件设备也需要来自 Nearby Interaction 的配置数据才能知道如何配置自己。此数据需要采用名为“可共享配置数据”方式。当您使用 Configuration 运行会话时，Nearby Interaction 将通过代理让你的应用程序提供可共享的配置数据。就像我们使用数据通道接收配件的配置数据一样，在这里，我们将再次使用数据通道将共享的配置数据发送回配件。代码如下：

```
func session(_ session: NISession,didGenerateShareableConfigurationData: Data,
for object: NINearbyObject) {
    // 通过工具方法 获取数据链接
    guard let conn = getConnection(object: object) else { return }
    //发送共享数据
    conn.sendSharedableConfigurationData(data)
}
```

didGenerateShareableConfigurationData 是 iOS 15 中的新回调。该回调提供了可共享的配置数据，并指示它应该去哪个硬件设备，这在与多个硬件设备交互时非常有用。应尽快的通过数据通道将数据发送到硬件。一般而言，管理与不同硬件的数据连接可以采用多种不同的形式，这一切都取决于需求。为简单起见，假设在我的应用程序中，我选择让每个与我交互的配件保持独立的数据连接。为了让我的代码看起来井井有条，我定义了一个工具函数，它根据我给它的 NearbyObject 返回给我一个连接。获得对连接的引用后，我将使用它将可共享配置数据发送到硬件设备。

part4 超时处理

如果 ShareableConfigurationData 发送得不够快，您的会话可能会超时。代码如下：

```
func session(_ session: NISession, didRemove nearbyObjects: [NINearbyObject], re
ason: NINearbyObject.RemovalReason) {
    // 只有超时才进行后续操作
    guard reason == .timeout else { return }
    // 获取硬件设备
    guard let accessory = nearbyObjects.first else { return }
    //是否应该重试 是否config 还在有效的状态
    if shouldRetryWithCachedConfig(accessory) {
        if let config = session.configuration {
            session.run(config)
        }
    }
}
```

与硬件的会话超时将通过 didRemove 的代理返回给应用程序。当 Nearby Interaction 给我 didRemove 的回调时，我将首先检查移除的原因。如果原因是 timeout，并且我非常确信硬件设备可能仍在附近，可以尝试进行重联。为了决定此附件是否应该进入“重试流程”，我定义了一个辅助函数，其中包含帮助我做出此决定的专门逻辑。诸如“我重试了多少次没有成功？”这样的情况。或“配件是否通知我它已停止？”或其他类似问题可以成为这个决策函数中的一部分。如果我决定重试，我所要做的就是使用相同的配置再次运行会话。请记住，缓存配置仅在硬件设备上的的会话未终止时才保持有效。如果会话终止的话就需要重新建立连接了。请记住，硬件设备上的会话是设备上运行的代码必须管理的事情，并且可以通过多种不同方式完成，所有这些都取决于应用的场景。

part5 处理数据的更新

接收到可共享的配置数据后，配件上的超宽带硬件将立即开始以适当的配置运行，以便与应用程序中的 NISession 进行交互。如果配件和运行应用程序的 iPhone 彼此靠近，会话将开始为您的应用程序提供 NearbyObject 更新流，其中包含距离和到硬件的方向。甚至可以通过为每个硬件创建和运行会话来同时与多个硬件设备互动。根据配件上的硬件功能，您还可以获得等效的邻近更新在配件上运行的代码中。从框架中获得 NearbyObject 更新后，您将如何处理它们？提醒一下，我们希望构建这样一种体验：当用户进入较大的区域时，应用程序和配件启用功能 A， 而当用户进入配件周围的较小区域时启用功能 B。代码如下：

```
func session(_ session: NISession, didUpdate nearbyObjects: [NINearbyObject]){
    guard let accessory = nearbyObjects.first else { return }
    guard let distance = accessory.distance else { return }
    let smoothedDistance = getSmoothedDistance(distance)

    if smoothedDistance < 1.5 {
        doA(smoothedDistance);
    } else if smoothedDistance < 3.0 {
        doB(smoothedDistance)
    }
}
```

代码中展示了如何在 iOS 应用程序中使用 NearbyObject 更新来执行此操作。当应用程序和硬件之间的会话正在运行时，有关硬件的更新将通过 didUpdate 进行回调。首先，我将获取框架为我们提供的附近对象的引用。接下来，我将创建一个带有到该对象的距离的局部变量，框架以米为单位提供距硬件的距离。接下来我要做的是将这些数据提供给我的应用程序中名为 getSmoothedDistance 的工具函数。我在我的应用程序中定义了这个函数来帮助我防范处理距离的各种异常。例如，在用户突然移动时，或者他们恰好站在区域之间的边界上。最后，我可以检查用户与配件的距离是否超过了我预定义的阈值。来根据当前距离选择启用 Function A 或 Function B。

总结

Nearby Interaction 在 iOS 14 上还只能是手机与手机之间进行交互，但在 iOS 15 上扩展到了手机与第三方的硬件。不断增强了 Nearby Interaction 框架的能力。后续在智能家具，AR/MR，室内定位，钱包，地铁检票都应该都会有广泛的应用。

关注我们

我们是「老司机技术周报」，一个持续追求精品 iOS 内容的技术公众号。欢迎关注。

老司机技术周报

微信扫描二维码，关注我的公众号

关注有礼，关注【老司机技术周报】，回复「WWDC」，领取《WWDC20 内参》

这个作品真棒，我要支持一下！

赞赏

赞 收藏 分享 转载

WWDC21 内参
关注「老司机技术周报」公众号，回复「2022」、「2021」免费领取往年内参系列 一年一度的 WWDC 又来啦！...
已经订阅

1条评论

老司机技术周报 #1
对文中有疑问的可以加我们的微信 iTDriverr，备注「WWDC21」拉你进入读者群交流哦。
7 月前 赞

写下你的评论