Allowing apps and websites to link to your content | Apple Developer Documentation

Archived: 2026-04-05 14:14:24 UTC

Overview

You can connect to content deep inside your app with universal links. Users open your app in a specified context, allowing them to accomplish their goals efficiently.

When users tap or click a universal link, the system redirects the link directly to your app without routing through the person's default web browser or your website. In addition, because universal links are standard HTTP or HTTPS links, one URL works for both your website and your app. If the person hasn't installed your app, the system opens the URL in their default web browser, allowing your website to handle it.

When someone installs your app, the system checks a file stored on your web server to verify that your website allows your app to open URLs on its behalf. Only you can store this file on your server, securing the association of your website and your app.

Support universal links

Take the following steps to support universal links:

1. Create a two-way association between your app and your website and specify the URLs that your app handles, as described in Supporting associated domains.
2. Update your app delegate to respond to the user activity object the system provides when a universal link routes to your app, as described in Supporting universal links in your app.

With universal links, users open your app when they click links to your website within their browser app and `WKWebView`, and when they click links that result in a call to:

- `open(_:options:completionHandler:)` in iOS and tvOS
- `openSystemURL(_:)` in watchOS
- `open(_:withApplicationAt:configuration:completionHandler:)` in macOS
- `openURL` in SwiftUI

When a user browses your website in Safari and taps a universal link in the same domain, the system opens that link in Safari, respecting the user's most likely intent to continue within the browser. If the user taps a universal link in a different domain, the system opens the link in your app.

Communicate with other apps

Apps can communicate through universal links. Supporting universal links allows other apps to send small amounts of data directly to your app without using a third-party server.

Define the parameters that your app handles within the URL query string. The following example code for a photo library app specifies parameters that include the name of an album and the index of a photo to display.

```
 https://myphotoapp.example.com/albums?albumname=vacation&index=1
https://myphotoapp.example.com/albums?albumname=wedding&index=17
```

Other apps craft URLs based on your domain, path, and parameters and ask your app to open them by calling:

- The `open( :options:completionHandler:)` method of `UIApplication` in iOS and tvOS
- The `openSystemURL( :)` method of `WKExtension` in watchOS
- The `open( :withApplicationAt:configuration:completionHandler:)` method of `NSWorkspace` in macOS
- The `openURL` environment value in SwiftUI

The calling app can ask the system to inform it when your app opens the URL.

In this example code, an app calls your universal link in iOS and tvOS:

```
 if let appURL = URL(string: "https://myphotoapp.example.com/albums?albumname=vacation&index=1") {
    UIApplication.shared.open(appURL) { success in
        if success {
            print("The URL was delivered successfully.")
        } else {
            print("The URL failed to open.")
        }
    }
} else {
    print("Invalid URL specified.")
}
```

In this example code, an app calls your universal link in watchOS:

```
 if let appURL = URL(string: "https://myphotoapp.example.com/albums?albumname=vacation&index=1") {
    WKExtension.shared().openSystemURL(appURL)
} else {
    print("Invalid URL specified.")
}
```

In this example code, an app calls your universal link in macOS:

```
 if let appURL = URL(string: "https://myphotoapp.example.com/albums?albumname=vacation&index=1") {
    let configuration = NSWorkspace.OpenConfiguration()
    NSWorkspace.shared.open(appURL, configuration: configuration) { (app, error) in
        guard error == nil else {
            print("The URL failed to open.")
            return
        }
        print("The URL was delivered successfully.")
    }
} else {
    print("Invalid URL specified.")
}
```

For more information on handling links within your app, see Supporting universal links in your app.

Source: https://developer.apple.com/ios/universal-links/