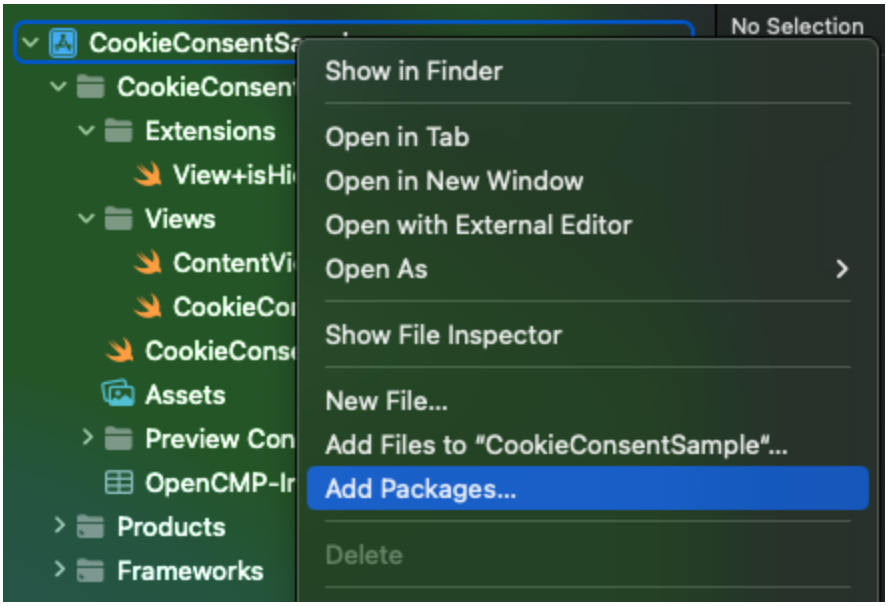
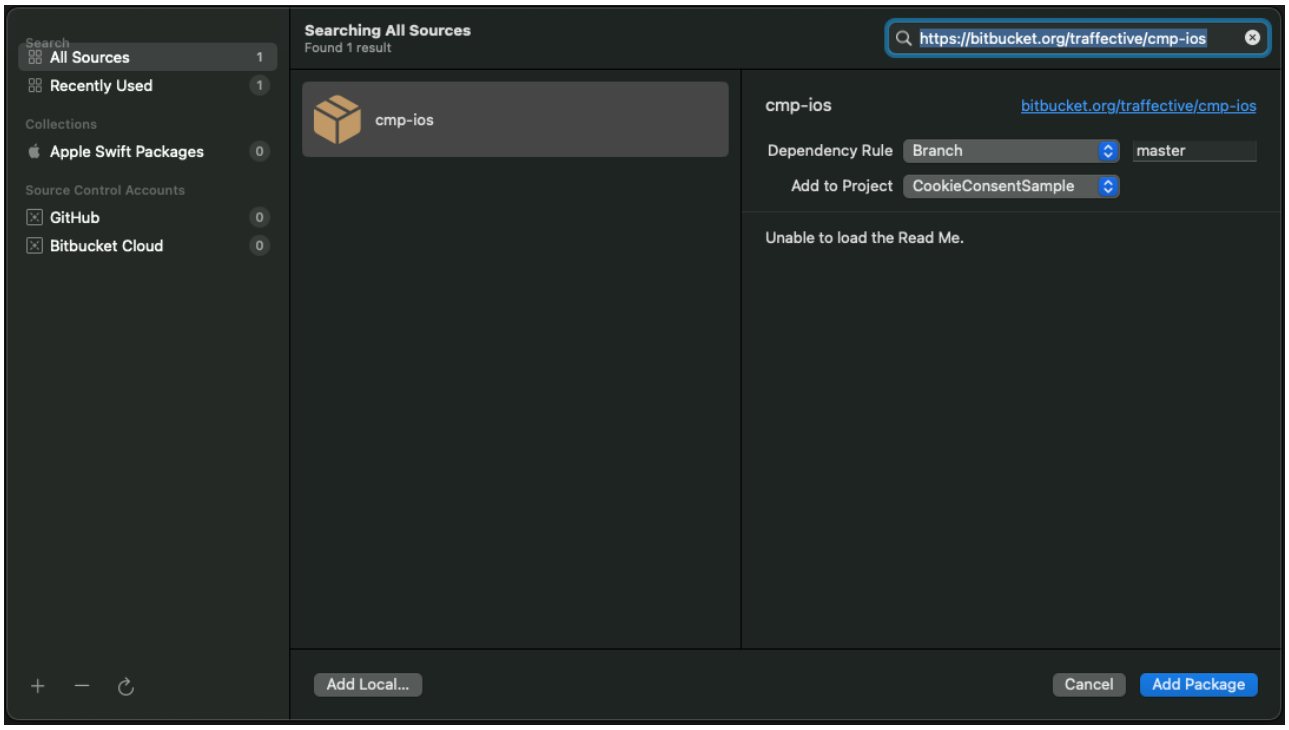
Open CMP iOS

Getting Started

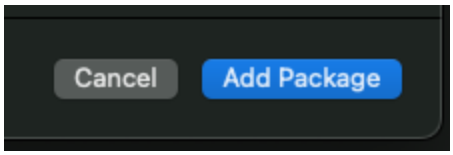
1. You must sign up on github.com to download the Open CMP Swift Package
2. Open Xcode and navigate to your iOS app project, as below click “Add Packages...”



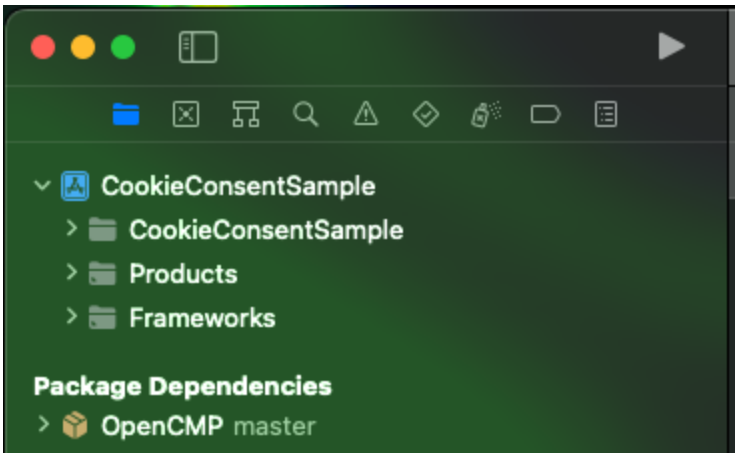
3. Our Open CMP package is publicly available. Therefore, with your GitHub account you have access to
4. Hooray! You are ready to download the Swift Package. You should navigate to “All Sources” or “Recently Used” to get to the search field on the top right, as below
5. Copy and paste the url https://github.com/OpenCMP-net/cmp-ios into the search field. You will get the ***cmp-ios*** Swift Package listed in the window



8. Now hit the button ***Add Package*** to finally add the new package to your Xcode project

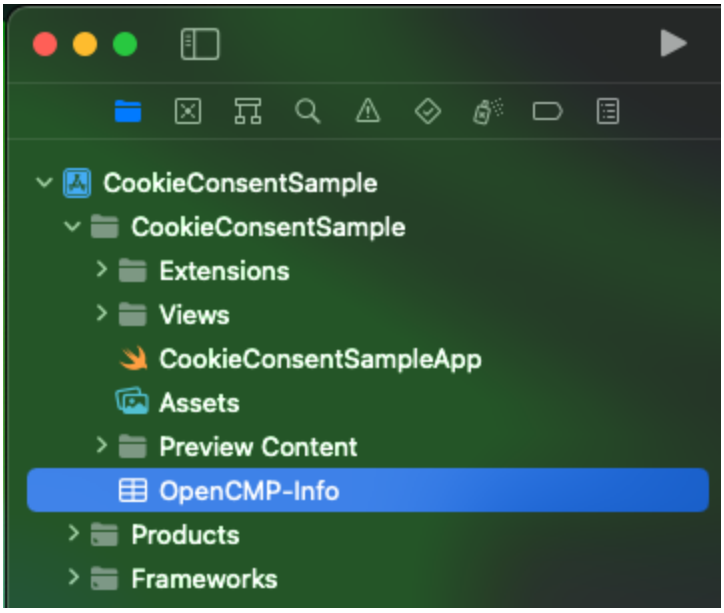


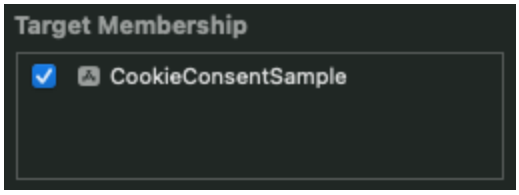
9. The Open CMP Swift Package is listed in the Xcode navigator, as below



Apply configuration

You can start with the default configuration below. However, we provide you with a customized configuration that you can use to apply your styling, etc. You need to copy the configuration file **OpenCMP-Info.plist** to the root directory.






```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
        <key>domain</key>
        <string>traffective.com</string>
```

```
        <key>allowedDomains</key>
        <array>
                <string>cdn.opencmp.net</string>
                <string>www.traffective.com</string>
        </array>
        <key>disableDomainAccessPolicy</key>
        <true/>
        <key>printBrowserLogs</key>
        <false/>
</dict>
</plist>
```

Apply remote configuration

You can inject the config object without using the **OpenCMP-Info.plist**. You should not have the **OpenCMP-Info.plist** in place if you inject the config object otherwise your config gets overwritten with the config from **OpenCMP-Info.plist** the file.

```
let config = OpenCMPConfig(
  domain: "traffective.com",
  allowedDomains: ["www.traffective.com"],
  disableDomainAccessPolicy: true,
  printBrowserLogs: false
)

let uiView = try? OpenCMPView.shared(config: config, acceptOrR
eject: acceptOrReject, showUi: showUi, hideUi: hideUi)
return (uiView!)
```

UIKit integration

Our Open CMP framework is developed in UIKit. Therefore it becomes easy to add the view of the Cookie Consent Banner into your app code. You will receive the cookie data via the ***acceptOrReject*** closure. It will dismiss the Open CMP view whenever the user accepted or rejected the consent.

Hint

> *Sometimes it is a challenge to further process data returned in a closure. We suggest to work with the* **Combine** *framework.*

```
let uiView = try? OpenCMPView.shared(acceptOrReject: { cookies in
  // You will receive the stored 'cookies' here.
}, showUi: { }, hideUi: { })
addSubview(uiView)
```

Show the consent ui again if a user clicks a button

```
OpenCMPView.shared?.showUiByUser()
```

Hint

> *You will get the new cookie consent again in the 'acceptOrReject' closure.*

Sample cookie data

The *cookie object* is strongly typed. Therefore it suggest you all the attributes of the *cookie object*. We also give you a sample of the possible cookie data:

```
tcf : "CPaXqOpPaXqPbAVACAENCTCsAP_AAH_AAAYgIZtV7T9cbGlDOX59YNtkWIUH1lAFouQCCBaAE6ABwCOAcKQEw2ASIEzoACACIBgAoDKBIAAEGEEQAEAQQIgBADHgIgiEhAAKIAJAABMBAQAAAAsKAAAAEAAIhEAZIACAmCqgQg5mQkAEIIAQQgABAAgAAKABAoMABAEIABgAAAAAgQAAAAAAAIIZgEiCpcQAFCGGBJIEEECIEQVhAFAKACCACAAiAAAAAAYKQAUUACAAAIAAAAAAAAAoAYBAAAAAEgAAEAAQIAAABAAAAAAAAAIAAJAABABAAAAAAgCAAAAAAAAAAAAAAAAAgAAQAAQCQEAAEAAQAgABAAAAAIAAAAAAAAAIAAAAAAAAAAAAAAAAAIAAAA.YAAAAAAAAAAA"
google : nil
```

```
custom : "CPaXqOpPaXqPbAVACAENCTCgAAAAAAAAAAAAFBwAgAWgKDAAAAA
A.YAAAAAAAAAAA"
meta : ConsentMetaData
  dv : 297
preferences : ConsentPreferences
  cmpSdkId : 21
  cmpSdkVersion : 2
  policyVersion : 2
  gdprApplies : 1
  publisherCc : "DE"
  useNonStandardStacks : 0
  vendorConsents : "11010101011110110100111111010111000110110000
01101001010000110011100101111110011111101011000001101101100100
00010110001000010100000111110101100101000000000101101000101110
01000000010000010000001011010000000001001110100000000000011100
00000001000111000000001110000101001000000010011000011011000000
00010010001000000100110011101000000000000010000000000010001000
00000110000000000010100000011001010000001001000000000000000000
01000001100001000001000100000000000010000000001000001000000010
00100000000001000000000011000111100000001000100000100010000100
10000100000000000000101000100000000000100100000000000000000100
11000000100000001000000000000000000000000000101100001010000000
00000000000000000000001000000000000000010001000010001000000000
01100100100000000000001000000010011000001010101010000001000010
00001110011001100100001001000000000001000010000010000000000100
00010000100000000000000010000000000010000000000000000000001010
00000000001000000101000001100000000000010000000010000100000000
00000000011000000000000000000000000000000000010000001000000000
00000000000000000000000000000000000001"
  vendorLegitimateInterests : "00000001001000100000101010010111
00010000000000000101000010000110000110000001001001001000000010
00001000001000000100010000001000100000101011000010000000001010
00000001010000000000010000010000000000010000000000000100010000
00000000000000000000000000000000011000001010010000000000001010
00101000000000001000000000000000000010000000000000000000000000
```

```
00000000000000000000000000001010000000001100000001000000000
00000000000000000000000100100000000000000000100000000000000
00100000100000000000000000000000001000000000000000000000000
00000000000000000000000000001000000000000000100100000000000
00000001000000000001000000000000000000000000000000000010000
00001000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000100000000000000000010
00000000000000100000001001000000100000000000000010000000000
00000001000000000100000000000000010000000000000000000000000
00000010000000000000000000000000000000000000000000000000000
00001000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000001"
  purposeConsents : "1111111111"
  purposeLegitimateInterests : "0111111111"
  specialFeaturesOptIns : "11"
  publisherConsent : ""
  publisherLegitimateInterests : ""
  publisherCustomPurposesConsents : ""
  publisherCustomPurposesLegitimateInterests : ""
  customVendorConsents : "00000000000000000000000000000000000
00000001000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000001"
  customVendorLegitimateInterests : ""
```

SwiftUI: Interfacing with UIKit (optional)

Our Open CMP framework is developed in UIKit. No worries! If you use SwiftUI you can easily develop a view class by using the ***Coordinator*** and ***UIViewControllerRepresentable*** interfaces.

```
import SwiftUI
import OpenCMP
import WebKit

struct CookieConsentView: UIViewControllerRepresentable {
  var acceptOrReject: (_ cookies: ConsentCookies?) -> Void

  var showUi: () -> Void

  var hideUi: () -> Void

  func makeUIViewController(context: Context) -> some UIViewCo
ntroller {
    let uiView = try? OpenCMPView.shared(acceptOrReject: accep
tOrReject, showUi: showUi, hideUi: hideUi)
    return (uiView!)
  }

  func updateUIViewController(_ uiViewController: UIViewContro
llerType, context: Context) {

  }

  func makeCoordinator() -> Coordinator {
    Coordinator(self)
  }
```

```
  internal class Coordinator {
    let cookieConsent: CookieConsentView

    init(_ cookieConsent: CookieConsentView) {
      self.cookieConsent = cookieConsent
    }
  }
}
```

SwiftUI integration sample code

```
import SwiftUI
import OpenCMP

struct ContentView: View {
  var body: some View {
    CookieConsentView(
      acceptOrReject: { cookies in print(cookies) },
      showUi: {  },
      hideUi: {  }
    )
  }
}
```

Show the consent ui again if a user clicks a button

```
Button("Show Ui", action: {
  OpenCMPView.shared?.showUiByUser()
})
```

Hint

> *You will get the new cookie consent again in the 'acceptOrReject' closure.*