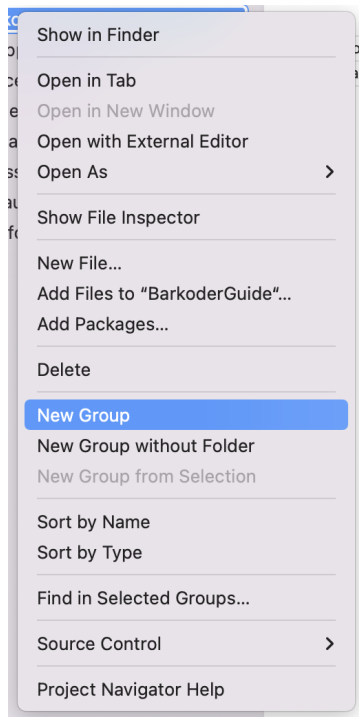


Barkoder Guide for iOS

Installation guide

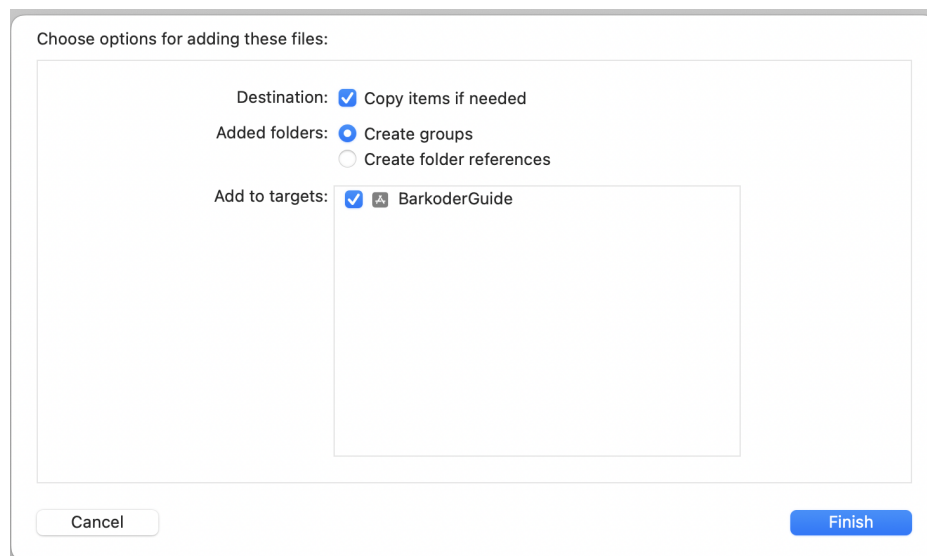
Please follow these simple steps to integrate our SDK into your iOS project

1. Create new group and name it “frameworks” (optional)

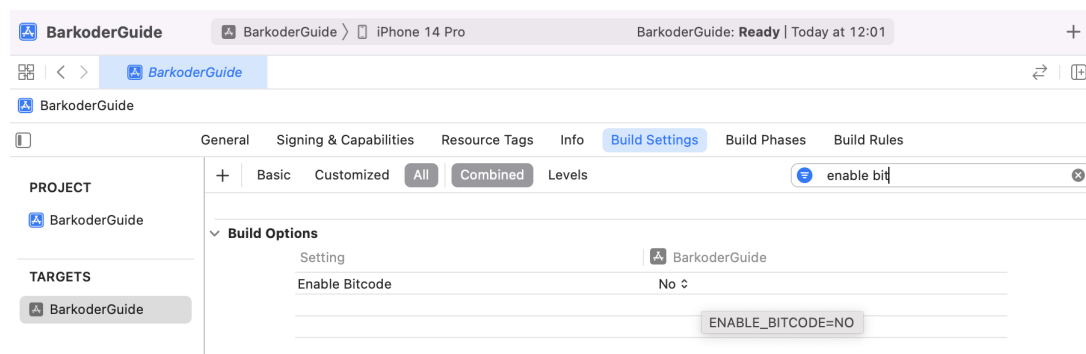


2. Add **Barkoder.xcframework** and **BarkoderSDK.xcframework** into frameworks

**Copy items if needed, Create groups and Add to desired targets)*



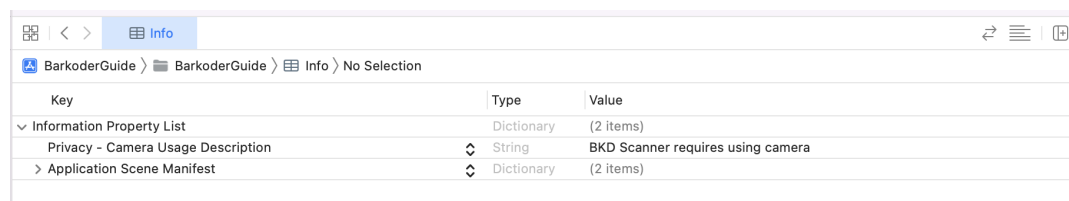
3. In **Build Settings** set **Enable Bitcode** to **NO**



4. Open the **Info.plist** with Source Code and add these changes

```
<key>NSCameraUsageDescription</key>  
<string>BKD Scanner requires using camera</string>
```

** Or using **Property List** add “Privacy - Camera Usage Description” and key “BKD Scanner requires using camera”*



5. Add **BarkoderView** as a view

```
import UIKit
import BarkoderSDK

class ViewController: UIViewController {

    private var barkoderView: BarkoderView!

    override func viewDidLoad() {
        super.viewDidLoad()
        setupUI()
        setupConstraints()
    }

    private func setupUI() {
        barkoderView = BarkoderView()

        view.addSubview(barkoderView)
    }

    private func setupConstraints() {
        barkoderView.translatesAutoresizingMaskIntoConstraints = false
        barkoderView.leadingAnchor.constraint(equalTo: view.leadingAnchor).isActive = true
        barkoderView.topAnchor.constraint(equalTo: view.topAnchor).isActive = true
        barkoderView.trailingAnchor.constraint(equalTo: view.trailingAnchor).isActive = true
        barkoderView.bottomAnchor.constraint(equalTo: view.bottomAnchor).isActive = true
    }
}
```

6. Create **Barkoder Config** per your needs

```
private func createBarkoderConfig() {
    // In order to perform scanning, config property need to be set before
    // If license key is not valid you will receive results with asterisks inside
    barkoderView.config = BarkoderConfig(licenseKey: "LICENSE_KEY") { licenseResult in
        print("Licensing SDK: \(licenseResult)")
    }

    // Enable QR barcode type
    guard let decoderConfig = barkoderView.config?.decoderConfig else { return }
    decoderConfig.qr.enabled = true
}
```

7. Implement **BarkoderResultDelegate** protocol where you will receive scanned results

```
extension ViewController: BarkoderResultDelegate {  
  
    func scanningFinished(_ decoderResults: [DecoderResult], thumbnails: [UIImage]?, image: UIImage?) {  
        if let textualData = decoderResults[0].textualData {  
            print("Scanned result: ", textualData)  
        }  
    }  
}
```

8. Start the scanning process

```
try? barkoderView.startScanning(self)
```

API description

BarkoderView

/// Set camera frames callback if you want to receive the frames/images only without decoding them and do your own work with the frames

@objc public func setPreviewFramesDelegate(_ delegate: BarkoderPreviewFramesDelegate?)

/// Set zoom factor for the camera preview

/// If preview session is already active this zoom factor will be set only for this session, therewise initial zoom will be set

/// Every next preview session will be started with this zoom factor

@objc public func setZoomFactor(_ zoomFactor: Float)

/// Check if current mobile device has flash available

@objc public func isFlashAvailable(_ completion: @escaping(_ flashAvailable: Bool) -> Void)

/// Turn flash ON/OFF

/// If preview session is already active this state be set only for active session

/// otherwise the initial flash state is set. Every next preview session will be started with this state

/// - Parameter enabled: [true, false]. Default value is false

@objc public func setFlash(_ enabled: Bool)

/// Start the camera preview only, without decoding

@objc public func startCamera()

/// Start the camera preview (if is not already running) and the scanning process

/// - **Parameter** resultDelegate

/// - **Throws** Error if BarkoderView config is not set

@objc public func startScanning(_ resultDelegate: BarkoderResultDelegate) **throws**

/// Stop the scanning process and the camera preview

@objc public func stopScanning()

/// Pause only the scanning process. Camera preview is still running

@objc public func pauseScanning()

BarkoderConfig

/// Get the decoder config object. With this object you can enable/disable decoders (barcode types) or configure each one of them

@objc public var decoderConfig: Config? = **nil**

/// Location line color as UIColor,

@objc public var locationLineColor: UIColor

/// Get the location line width as float

/// Default value is 2.0

@objc public var locationLineWidth: Float

/// Region of interest line color as UIColor

@objc public var roiLineColor: UIColor

/// Region of interest line width as float

/// Default value is 2.0

@objc public var roiLineWidth: Float

/// Region of interest background color as UIColor

@objc public var roiOverlayBackgroundColor: UIColor

/// Check if camera preview session will be closed when barcode is scanned

/// Default value is true

@objc public var closeSessionOnResultEnabled: Bool

/// Check if the image result is enabled

/// Image result is received in BarkoderResultDelegate as UIImage

/// Default value is false

@objc public var imageResultEnabled: Bool

/// Check if barcode location in the image result is enabled

/// If enabled, barcode in the result image will be marked

/// Default value is false

@objc public var locationInImageResultEnabled: Bool

/// Check if barcode location in preview is enabled

/// Default value is true

@objc public var locationInPreviewEnabled: Bool

/// Set active region of interest

@objc open func setRegionOfInterest(_ value: CGRect) throws

/// Get active region of interest

/// Default value is 'CGRect(x: 3, y: 30, width: 94, height: 40)'

@objc open func getRegionOfInterest() -> CGRect

/// Set maximum threads that will be used for the decoding process

/// - **Parameter** value: [1, max threads available]

/// - **Throws** Error if input param is greater than maximum threads available on that device

@objc func setThreadsLimit(_ value: Int) throws

/// Check if the camera preview can be zoomed with pinch

/// Default value is false

@objc public var pinchToZoomEnabled: Bool

/// Check if ROI is visible on the preview screen

/// Default value is true

@objc public var regionOfInterestVisible: Bool

/// Get the active resolution. It can be Normal(HD), or HIGH(Full HD)

/// Default value is BarkoderView.BarkoderResolution.normal

@objc public var barkoderResolution: BarkoderView.BarkoderResolution

/// Check if device will beep on successful scan

/// Default value is true

@objc public var beepOnSuccessEnabled: Bool

/// Check if device will vibrate on successful scan

/// Default value is true

@objc public var vibrateOnSuccessEnabled: Bool

```

/// Getting barcode thumbnail on result
/// Default value is true
@objc public var barcodeThumbnailOnResult: Bool

/// Getting threshold between duplicates scans
/// Default value is 5
@objc public var thresholdBetweenDuplicatesScans: Int

```

BarkoderHelper

```

/// Scan barcode from bitmap image
/// - Parameters:
///   - image: Image that you want to be scanned as UIImage
///   - bkdConfig: config that will be used for scanning process
///   - resultDelegate: where you will receive scanned result
@objc public static func scanImage(_ image: UIImage, bkdConfig: BarkoderConfig,
resultDelegate: BarkoderResultDelegate)

/// Apply config params from predefined template
/// - Parameters:
///   - config: that will be configured
///   - template: that will be applied on config
///   - finished: that will be executed when this function is finished
@objc public static func applyConfigSettingsFromTemplate(_ config: BarkoderConfig,
template: BarkoderConfigTemplate, finished: @escaping (BarkoderConfig) -> Void)

/// Retrieve config properties from the URL and apply them in the config that is send as input
param
/// - Parameters:
///   - config: that will be configured
///   - url: URL to the JSON file
///   - finished: callback that will be executed when this function is finished
@objc public static func applyConfigSettingsFromURL(_ config: BarkoderConfig, url: URL,
finished: @escaping (BarkoderConfig?, Error?) -> Void)

/// Retrieve config properties from the URL and apply them in the config that is send as input
param
/// - Parameters:
///   - config: that will be configured
///   - url: filePath to the JSON file
///   - finished: callback that will be executed when this function is finished
@objc public static func applyConfigSettingsFromFile(_ config: BarkoderConfig, url: String,
finished: @escaping (BarkoderConfig?, Error?) -> Void)

```

```
/// Export config that is send as input param to JSON string  
/// - Parameter barkoderConfig: config that will be exported  
/// - Returns: JSON string  
@objc public static func configToJSON(_ barkoderConfig: BarkoderConfig) -> String?
```