Instructions for integrating Scannr into your app

You can integrate with Scannr by openning it from your app and have the result of the scan be returned to your app.

1. For Android applications

You can use a Scannr URL scheme to launch Scannr and perform ID scanning to obtain data from drivers licence. URL scheme that allows you to launch Scannr app is:

```
scannr://
```

For example, you can launch Scannr by following code:

```
private val SCAN_RESULTS_REQUEST = 42  // The request code

val scannrIntent = Intent(Intent.ACTION_PICK, Uri.parse("scannr://"))
startActivityForResult(scannrIntent, SCAN_RESULTS_REQUEST)
```

Note: the code is in Kotlin.

Now you need to implement `onActivityResult` method. In it you will receive scan results. Put this code in the same activity where you called `startActivityForResult` method and implement todo part.

```
override fun onActivityResult(requestCode: Int, resultCode: Int, data:
Intent?) {
    when(requestCode) {
        SCAN_RESULTS_REQUEST)-> {
            if (resultCode == Activity.RESULT_OK) {
                scanData = data?.getStringExtra("scanData")

                if (scanData != null) {
                   //todo do what you need to do with scan
                }
            }
        }
    }
}
```

The string you will get from Scannr will be in json format. This is an example of the data from Scannr:

```
{
```

```
    "scanning_timestamp": 1421336230,
    "scanning_result": {
    "kPPDateOfBirth": "01012000",
    "kPPHeightCm": "180",
    "kPPWeightKilograms": "70",
    "kPPCustomerMiddleName": "A",
    "kPPCustomerIdNumber": "D01234567",
    "kPPOrganDonor": "Y",
    "kPPWeightPounds": "154",
    "kPPAamvaVersionNumber": "1",
    "kPPHeight": "68 IN",
    "kPPIssuerIdentificationNumber": "123456",
    "kPPDocumentIssueDate": "12345678",
    "kPPCustomerFamilyName": "DOE",
    "kPPAddressJurisdictionCode": "AA",
    "kPPAddressStreet": "1234 Address",
    "kPPNonResident": "N",
    "kPPJurisdictionVehicleClass": "A",
    "kPPIssuingJurisdiction": "AA",
    "kPPDocumentExpirationDate": "01012020",
    "kPPIssueTimestamp": "2001-12-01 00.00.00.000000",
    "kPPAddressCity": "CITY NAME",
    "kPPHairColor": "BR",
    "kPPHeightIn": "68",
    "kPPSex": "1",
    "kPPJurisdictionRestrictionCodes": "B",
    "kPPEyeColor": "GRN",
    "kPPJurisdictionVersionNumber": "0",
    "kPPCustomerFullName": "DOE,JOHN,A",
    "kPPAddressPostalCode": "12345-1234",
    "kPPCustomerFirstName": "JOHN"
    }
}
```

2. For iOS applications

You can use a Scannr URL scheme to launch Scannr and perform ID scanning to obtain data from drivers licence. URL scheme that allows you to launch Scannr app is:

```
scannr://?callbackScheme=<foo>
```

For example, you can launch Scanner by following code:

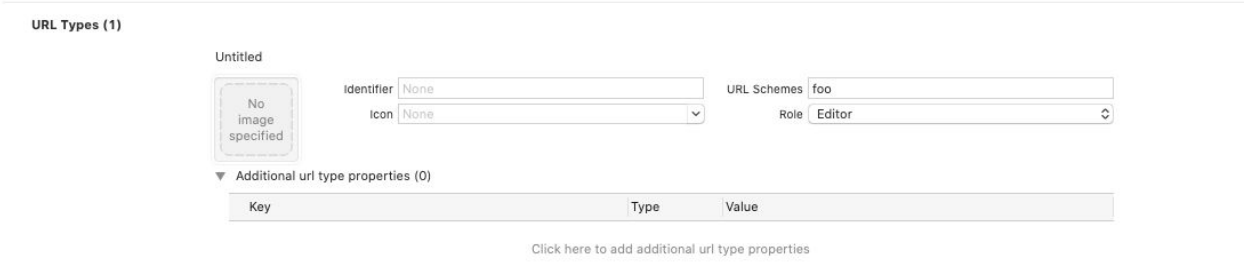
[Objective-C]

```
if ([[UIApplication sharedApplication] canOpenURL:[NSURL
URLWithString:@"scannr://?callbackScheme=<foo>"]])
{
   [[UIApplication sharedApplication] openURL:[NSURL
URLWithString:@"scannr://?callbackScheme=<foo>"] options:@{} completionHandler:nil];
}
else
{
   NSLog(@"Can't use Scannr, did you check your LSApplicationQueriesSchemes?");
}
```

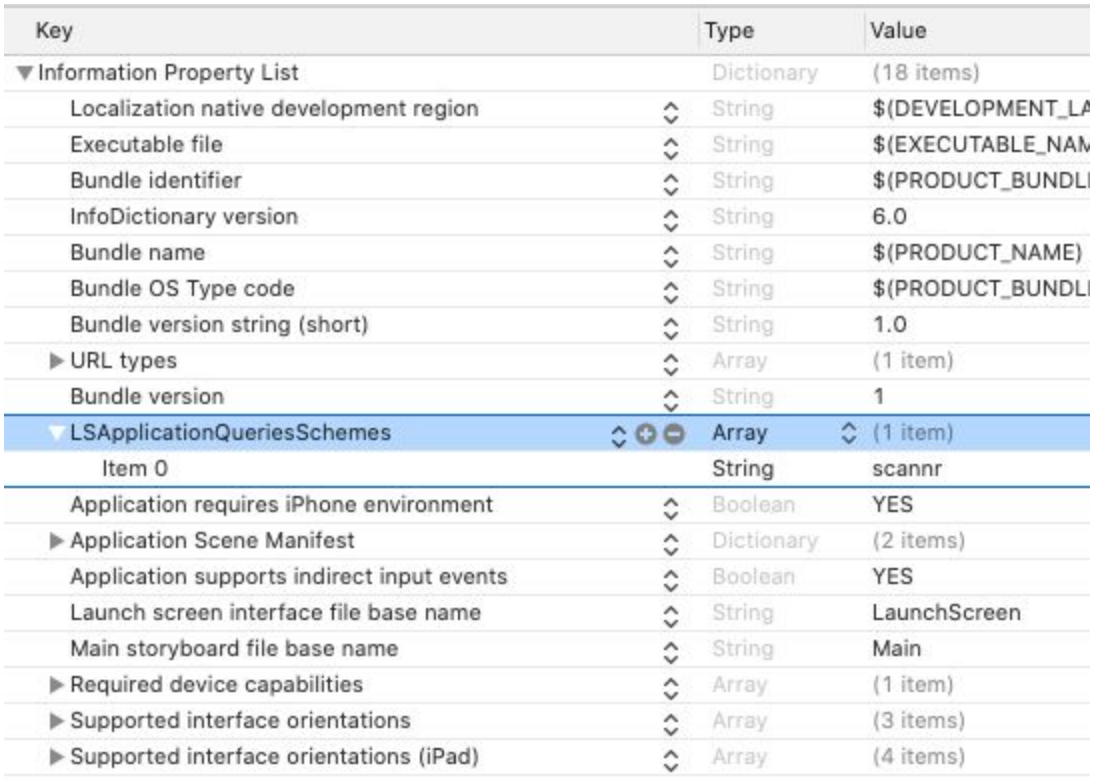
[Swift]

```
if let url = URL(string: "scannr://?callbackScheme=<foo>") {
    if UIApplication.shared.canOpenURL(url) {
        UIApplication.shared.open(url)
    } else {
        print("Can't use Scannr, did you check your LSApplicationQueriesSchemes?")
    }
}
```

Define **foo** URL scheme in your application project settings, under Info. The result is displayed on image:



You also have to whitelist LSApplicationQueriesSchemes values in your app target's info.plist. The result is displayed in the following image:



After that you have to implement a method to obtain the scanning result.
If you are using AppDelegate refer to this example:

[Objective-C]

```
- (BOOL)application:(UIApplication *)app
            openURL:(NSURL *)url
            options:(NSDictionary<UIApplicationOpenURLOptionsKey, id> *)options
{
   NSString *urlString = [url.absoluteString stringByRemovingPercentEncoding];
```

```
   NSString *resultString = [urlString
stringByReplacingOccurrencesOfString:@"<foo>://" withString:@""];
   NSError *error = nil;
   NSDictionary *resultDict = [NSJSONSerialization JSONObjectWithData: [resultString
dataUsingEncoding:NSUTF8StringEncoding] options:kNilOptions error:&error];
   if (!error)
   {
       NSLog(@"RESULT OF SCANNING: %@", resultDict);
   }
   else
   {
       NSLog(@"ERROR: %@", error);
   }
   return YES;
}
```

[Swift]

```
func application(_ app: UIApplication, open url: URL, options:
[UIApplication.OpenURLOptionsKey : Any] = [:]) -> Bool {
    let url = url.absoluteString
    let urlString = url.removingPercentEncoding
    let resultString = urlString?.replacingOccurrences(of: "<foo>://", with: "")
    guard let data = resultString?.data(using: String.Encoding.utf8) else {
        return false
    }
    do {
        let resultDict = try JSONSerialization.jsonObject(with: data, options: [])
        print("RESULT OF SCANNING: \(resultDict)")
    } catch let error as NSError {
        print("ERROR: \(error)")
    }
    return true
}
```

If you are using Scene Delegate refer to this example:

[Objective-C]

```
- (void)scene:(UIScene *)scene willConnectToSession:(UISceneSession *)session
options:(UISceneConnectionOptions *)connectionOptions {
   NSURL *url = connectionOptions.URLContexts.allObjects.firstObject.URL;
}

- (void)scene:(UIScene *)scene openURLContexts:(NSSet *)URLContexts {
   NSString *urlString = [[[URLContexts.allObjects.firstObject URL] absoluteString]
stringByRemovingPercentEncoding];
   NSString *resultString = [urlString
stringByReplacingOccurrencesOfString:@"<foo>://" withString:@""];
   NSError *error = nil;
   NSDictionary *resultDict = [NSJSONSerialization JSONObjectWithData: [resultString
dataUsingEncoding:NSUTF8StringEncoding] options:kNilOptions error:&error];
   if (!error)
   {
       NSLog(@"RESULT OF SCANNING: %@", resultDict);
   }
   else
   {
       NSLog(@"ERROR: %@", error);
   }
}
```

[Swift]

```
func scene(_ scene: UIScene, willConnectTo session: UISceneSession, options
connectionOptions: UIScene.ConnectionOptions) {
    guard let _ = (scene as? UIWindowScene) else { return }
    let url = connectionOptions.urlContexts.first?.url
}

func scene(_ scene: UIScene, openURLContexts URLContexts: Set<UIOpenURLContext>) {
    guard let url = URLContexts.first?.url.absoluteString else {
        return
    }
    let urlString = url.removingPercentEncoding
```

```
    let resultString = urlString?.replacingOccurrences(of: "<foo>://", with: "")
    guard let data = resultString?.data(using: String.Encoding.utf8) else {
        return
    }
    do {
        let resultDict = try JSONSerialization.jsonObject(with: data, options: [])
        print("RESULT OF SCANNING: \(resultDict)")
    } catch let error as NSError {
        print("ERROR: \(error)")
    }
}
```

Variable `resultDict` will contain the result of scanning. Here you can find more about keys and values contained in `resultDict`. Here you can find more about custom URL Schemes.