AsReaderDock and AsReaderGun React Native Wrapper Documentation

Table of Contents

Release Versions

1.0

Date of Release – 09/09/2022

1. Initial release of the document.

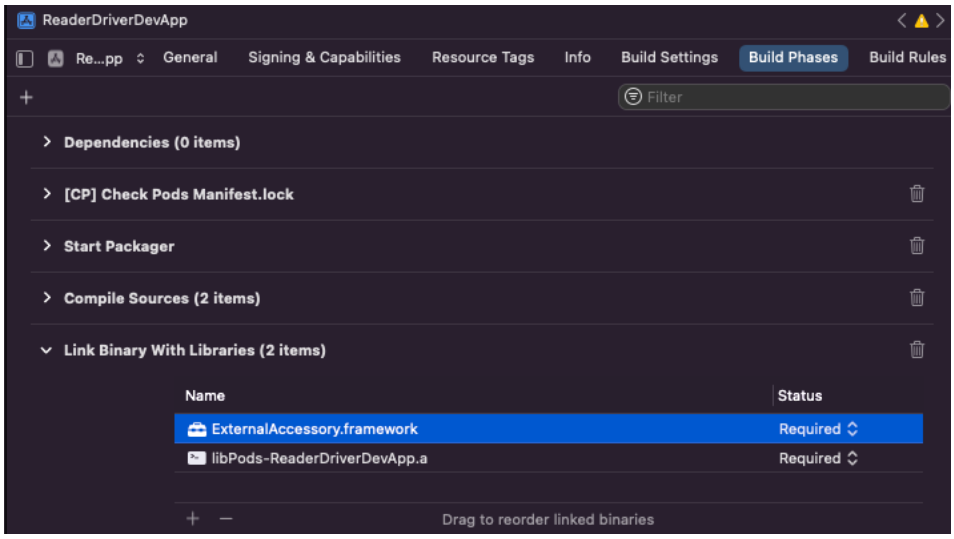
1.1

Date of Release – 03/30/2022

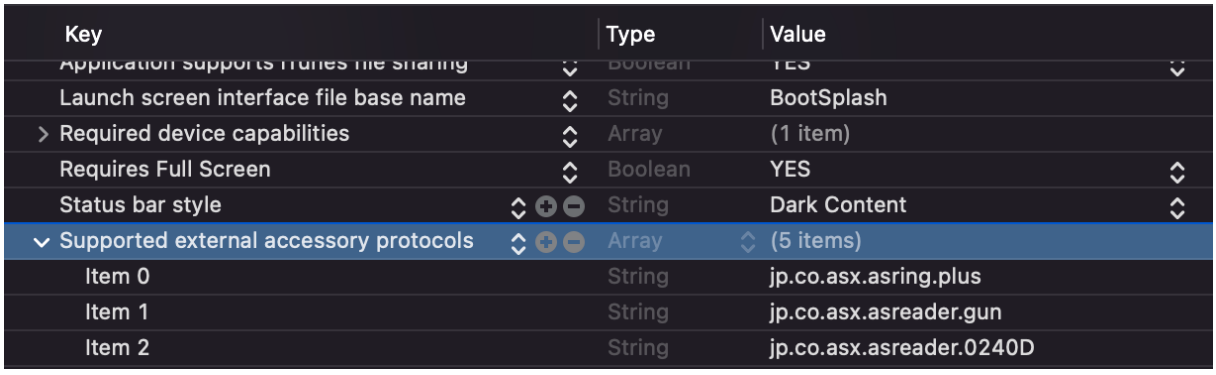
1. Add AsReaderGun wrapper

1. Setup

1. Upload the npm package ios-edge-driver using “npm publish”
2. Once published, in React Native projects where the package is being uses run “npm install ios-edge-driver”
3. In the ios directory of the project, run “pod install”
4. Open the project’s, .xcworkspace file in Xcode and go to the “Build Phases” page in Target settings. In “Link Binary with Libraries” add “ExternalAccessory.framework” (highlighted below).



5. In Xcode, open the project’s info.plist file. For key “Supported external accessory protocols”, for AsReaderDock, add “jp.co.asx.asreader.0240D” as an item. For AsReaderGun, add “jp.co.asx.asring.plus” and “jp.co.asx.asreader.gun”:



2. Usage

Accessing the Native Module:

```
import { NativeModules } from "react-native";
const { RNReaderModule } = NativeModules;

// Call methods on RNReaderModule, ie “RNReaderModule.getDeviceDetails();”
```

Creating event emitter:

```
import { NativeEventEmitter, NativeModules } from "react-native";
const { RNReaderModule } = NativeModules;

const events = new NativeEventEmitter(RNReaderModule);
```

Adding event listener example:

```
events.addListener("barcodeRead", (barcode) => {
  // process barcode
});
```

3. RNReaderModule Methods

- **getSupportedReaders() => Promise<vendor: string, model: string, modelType: string>**
- **getAvailable() => Promise<**
  {
  vendor: string,
  model: string,
  modelType: string,
  serialNumber: string
  }[]
  >
  - Returns a list of the readers that are plugged in or connected via Bluetooth and are available to be connected to
- **loadDriver(deviceName: string, model: string, modelType: string, inputType: 'usb' | 'bluetooth') => Promise<void>**
  - Loads the selected driver
  - vendor: name of the vendor of the reader
  - model: name of the model of the reader
  - modelType: name of the type of reader
  - inputType: type of input of the reader
  - throws LoadDriverError if unable to load driver because the device name was invalid
- **getDriverName() => Promise<string>**
  - Returns the name of the currently loaded driver
- **getDriverVersion() => Promise<{string: string}>**
  - Returns information about the version hierarchy of the driver
- **getDeviceDetails() => Promise<object>**
  - Returns JSON object with details of the device, like firmware, serial number, etc
- **connect(address: string) => Promise<void>**
  - Connects to the reader
  - address: the Bluetooth address of the reader if it is needed to connect. For non-Bluetooth readers, any string works
  - Throws ReaderNotFoundError
- **disconnect() => Promise<void>**
  - Disconnects from the reader
  - Throws ReaderNotFoundError
- **isConnected() => Promise<boolean>**
  - Returns the connection status of the reader
- **getBatteryPercentage() => Promise<number>**
  - Returns the battery percentage of the reader
  - Throws ReaderNotFoundError and ReaderConnectionError
- **switchToScanRfidMode() => Promise<void>**
  - Switches to RFID reading mode and starts reading tags if trigger pressed
  - Throws ReaderNotFoundError and ReaderConnectionError

- **startRfidScan() => Promise<void>**
  - Starts continuous RFID reading meaning that tags are read continuously without trigger needing to be pressed
  - Throws ReaderNotFoundError and ReaderConnectionError
- **stopRfidScan() => Promise<void>**
  - Stops continuous RFID reading
  - Throws ReaderNotFoundError and ReaderConnectionError
- **activateSearchMode(tagToFind: string) => Promise<void>**
  - Enables search mode with given tag EPC, so that only a tag with this EPC is sent
  - tagToFind: EPC of the tag to search for
  - Throws ReaderNotFoundError and ReaderConnectionError
- **stopReading() => Promise<void>**
  - Stops RFID or barcode rading
  - Throws ReaderNotFoundError and ReaderConnectionError
- **switchToScanBarcodeMode() => Promise<void>**
  - Switches to barcode scanning mode
  - Throws ReaderNotFoundError and ReaderConnectionError
- **getReaderPowerRange() => Promise<minPower: number, maxPower: number>**
  - gets the min and max power of the reader
- **getReaderPower() => Promise<powerPercentage: number>**
  - gets the current power of the reader as a percentage
  - Throws ReaderNotFoundError and ReaderConnectionError
- **configureReaderPower(powerPercentage: number) => Promise<void>**
  - set the power of the reader
  - powerPercentage: the percentage of the power to set the reader to
  - Throws ReaderNotFoundError and ReaderConnectionError
- **addBarcodeEventListener: (callback: (barcodes: string[]) => void) => EmitterSubscription**
  - Register an event for a barcode being scanned. Callback is called with a barcode as a string[]
  - callback: to call when a barcode is scanned
  - return an EmitterSubscription object for the attached listener
- **addRfidEventListener: (callback: (tags: Tag[]) => void) => EmitterSubscription**
  - Register an event for a RFID tag being read. Callback is called with a tag as a string
  - callback: to call when a RFID tag is scanned
  - return an EmitterSubscription object for the attached listener
  - Tag type:

    {

    "epc": the tag's EPC,

    "rssi": the tag's RSSI,

    "scaledRssi": percentage strength of the RSSI

    "tagCount": the number of times this tag was read,

    "tid": the tag's TID (currently always returns empty string for AsReaderDock)

    }

4. Errors

ReaderNotFoundError – This error occurs when the reader is not available, either because it is not plugged in or not connected via Bluetooth.
ReaderConnectionError – This error occurs if an operation which requires the reader to be connected (eg getDeviceDetails) is called before a successful call to connect.
NoDriverLoadedError – This error occurs if a method is called when no driver has been successfully loaded.
NoDriverLoadedError – This error there is an unsupported device name in call to loadDriver.

5. Example

The following example would connect to AsReaderDock, turn on RFID reading, and print any scanned tags

```
import { NativeEventEmitter, NativeModules } from "react-native";
const { RNReaderModule } = NativeModules;

const events = new NativeEventEmitter(RNReaderModule);
events.addRfidEventListener((tagData) => {
  console.log(tagData[0].epc, tagData[0].rssi);
});

await RNReaderModule.loadDriver("AsReader", “ASR-L251G”, “AsReaderGun”, “usb”);
await RNReaderModule.connect("");
await RNReaderModule.switchToScanRfidMode();
```