

Oracle Retail Store Operations Cloud Service (SOCS) Barcode Scanner Integration (Doc ID 2694964.1)

In this Document

[Abstract](#)[History](#)[Details](#)[Summary](#)

[Options Available When Mobile Application is Deployed on Android](#)

[Option A: Use wedge scanner \(Hardware solution w/ sticky focus flag enabled\)](#)[Option B: Use camera barcode scanner](#)[Option C: Using device specific scanning SDKs](#)[How to develop a custom plugin?](#)[How to register plugins?](#)[How to integrate custom Cordova plugin into MAF application?](#)

[Why is this a challenge on iOS?](#)

[Why not take same approach we took on Android?](#)

[Options available for iOS](#)

[Option A: Use wedge scanner \(Hardware solution w/ sticky focus flag enabled\)](#)[Option B: Use camera barcode scanner](#)[Option C: Customize code with specific vendor sled using objective-C](#)[How to develop a custom plugin?](#)[How to register plugins?](#)[How to integrate custom Cordova plugin into MAF application?](#)

[Options available for Universal Windows Platform](#)

[Use wedge-scanner \(either Bluetooth or wired – with sticky focus flag enabled\)](#)[Supported plugins](#)[Reconfiguring Integrated Scanner Plugin for Android](#)

APPLIES TO:

Oracle Retail Store Operations Cloud Service - Version 19.0 and later
Information in this document applies to any platform.

ABSTRACT

Disclaimer - The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, timing, and pricing of any features or functionality described for Oracle's products may change and remains at the sole discretion of Oracle Corporation.

This document provides details that would guide Retailer and their Solution Implementers on possible options to enable Barcode Scanner Integration with the Retail Store Operations Cloud Service MAF (Mobile Application Framework) application.

Store inventory operations cloud service (SIOCS) is an umbrella term that includes two (2) cloud services.

- 1 Enterprise Inventory Cloud service (EICS)
 - a. This is hosted in GBUCS 3
 - b. Includes JET browser based UI, Java EE Server on Oracle Database and other components
- 2 Store Operations Cloud Service (SOCS)
 - a. This is a mobile application
 - b. Built using Oracle MAF (Mobile Application Framework)
 - c. Downloaded from Oracle Software Delivery Cloud & built for Android, iOS and Windows 10

HISTORY

Created: 27-JUL-2020

Updated: 22-AUG-2020

DETAILS

SOCS has been tested and certified by Oracle Retail Development with Retail industry leading Zebra MC and TC series devices. These devices make use of

- Zebra Enterprise mobility development kit (EMDK) for Android
- Zebra DataWedge background service that communicates Scan hardware events to the SOCS application.

The background service is a separate application which must be installed on the hardware device.

Zebra EMDK for Android

Zebra Enterprise mobility development kit (EMDK) for Android includes comprehensive set of APIs including data capture. These APIs can be used for scanner and peripheral support.

Zebra DataWedge

DataWedge is a utility that allows advanced data capturing capability using barcode scanning, serial port communication and Magnetic Stripe Readers (MSR) to any application. It runs in the background and handles the interface to both built-in and attached barcode scanners, serial ports and MSR modules in Motorola mobile computers.

The DataWedge app is what does the low-level communication with the scanner. DataWedge then signals SOCS Mobile through an Android "Intent" (the app must be manually configured to send this intent to the SOCS Mobile application by creating a profile). The SOCS Cordova plug-in on Android listens for intents sent to it from DataWedge. When the Intent is received, it parses the scanner data out of it for use within the SOCS mobile application.

Android Intent

Android supports communication between two mobile apps via Intent (Shared space between apps). The benefit of this is that SOCS does not need to directly interface with the vendor SDK and hardware. Vendor app decodes the scanned barcode and puts it on Intent with a shared key. Oracle SOCS MAF Application reads the barcode information from Intent via a shared key.

SUMMARY

Options Available When Mobile Application is Deployed on Android

Scanning options when not using Zebra devices

Option A: Use wedge scanner (Hardware solution w/ sticky focus flag enabled)

Wedge scanners are generic scanner solutions that can be used without writing any custom code. They can be built-in/wired/bluetooth-enabled. The only caveat is they need to have focus enabled in a field in order to scan. SOCS has a system configuration option to enable '*Scan Focus Item Detail*' (focus stays on same field), so when a user selects an entry field all scans done by the scanner/wedge-scanner gets captured for that field.

Sticky Focus flag can be configured in *.adf\META-INF\adf-config.xml

Option B: Use camera barcode scanner

Camera scanning isn't really a fast-scan (scan-scan) option but will work for small fashion Retailers with low-volume of scanning. Downside for any camera-based scans can be the dependency on focus & ambient lighting situation.

Implementers could try the following steps as part of a custom configuration:

- Go to the MAF sample applications directory for your BarcodeDemo project (/jdev/extensions/oracle.maf/Samples/PublicSamples/BarcodeDemo/src) and copy the folder phonegap-plugin-barcodescanner-master to your SOCS Mobile MAA project dir/plugins.
- Rename the folder to phonegap-plugin-barcodescanner in your SOCS Mobile plugin folder.
- In your SOCS Mobile MAA project in JDeveloper, go to Application Resources > Descriptors > ADF META-INF and open maf-plugins.xml.
- Add the following cordova plugin entry:
- Go into adf-config.xml and set DEVICE_CAMERA_SCAN to true.
- Rebuild and redeploy the application.

Any device with a MAF-usable camera should now have a camera scan option on any Scan Bar inside the MAF Mobile application.

Option C: Using device specific scanning SDKs

Due to a fractured Android marketplace, Oracle Retail SOCS cannot bundle all types of Vendor SDKs available in the market. If the device doesn't support DataWedge and EMDK, then the Retailer needs to customize SOCS with a specific vendor SDK. With DataWedge and EMDK, it is easier to write custom software that works with specific EMDK. If your device's Android OS supports that EMDK, all the scanning would continue to work. Vendor SDK decodes the scanned barcode and passes it to the Cordova plugin which then communicates this information to SOCS MAF application.

More on Android Cordova plug-in can be found at <https://cordova.apache.org/docs/en/latest/guide/platforms/android/plugin.html>.

How to develop a custom plugin?

Please read the [Plugin Development Guide](#) for an overview of the plugin's structure and its common JavaScript interface. In addition to that also read the [Android Plugin Development Guide](#) which provides details on how to implement the native plugin code on the Android platform.

Finally, read the 'Introduction to custom Cordova Plugin Development' at <https://blogs.oracle.com/digitalassistant/post/introduction-to-custom-cordova-plugin-development> for information on creating a very simple plugin for Android and iOS. These documentation resources will help you develop and build your own plugin if needed.

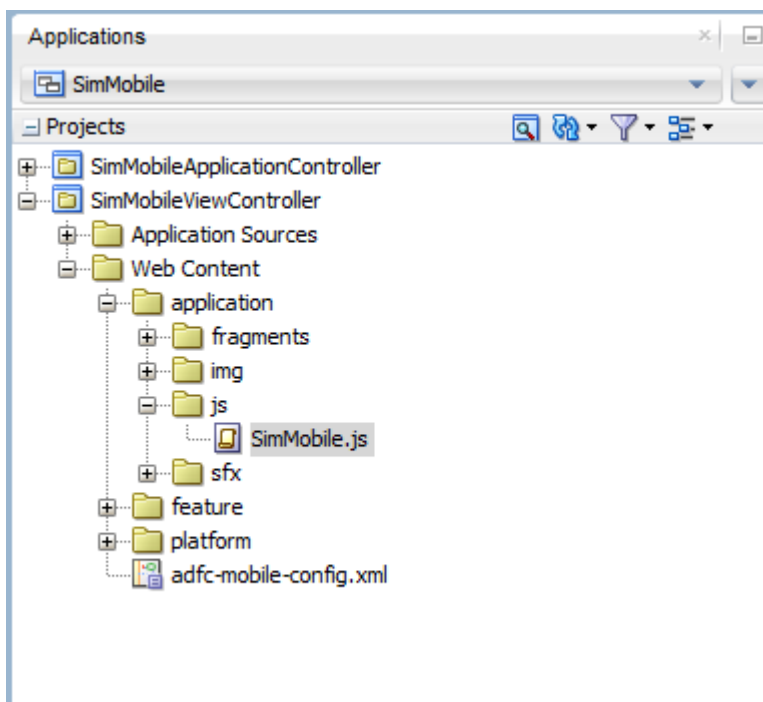
How to register plugins?

Implementers needs to register the additional scanner plugins in to the SOCS application. Review [How to Register an Additional Plugin](#) for MAF application for the steps. Before you begin, ensure that the SOCS application and the plugin to be registered with the application, are stored on the same drive. Or you can place the plugin in SOCS workspace/plugins directory with the other SOCS provided plugins.



How to integrate custom Cordova plugin into MAF application?

After registering the custom scanner plugin, the plugin needs to be invoked to use the barcode scanner application on the device. Read the 'Integrating a custom Cordova plugin into a MAF app' at https://blogs.oracle.com/digitalassistant/post/entry/integrating_a_custom_cordova_plugin for information about how you can invoke a plugin from Java, from a MAF AMX page, and from local HTML. In SOCS, the plugin methods are invoked in *SimMobile.js*, located as shown in below:



Code snippet from an existing RFID plugin invoking registerScanner method:

```
retail.sim.initializeSimRfidPlugin = function () {
    adf.mf.log.Framework.logp(adf.mf.log.level.FINE, "SimMobile.js",
    "initializeSimRfidPlugin", "Initializing SIM RFID plugin.");

    try {
        cordova.plugins.retail.sim.rfid.registerScanner(retail.sim._scanBarcodeSuccess,
        retail.sim._scanBarcodeError);
    }
}
```

```

    }

    catch (error) {

        adf.mf.log.Framework.logp(adf.mf.log.level.SEVERE, "SimMobile.js",
"initializeSimRfidPlugin", "Error initializing SIM RFID plugin: " + error);

    }}

```

Why is this a challenge on iOS?

The combination of various sled manufactures, 32-bit v/s 64-bit iPod / iPad mini /iPad devices, continuously updating iOS versions along with closed Apple systems, makes it challenging for a 3rd party Software Manufacturer (i.e. Oracle) to come up a generic scanner offering that is applicable to the broad market.

Why not take same approach we took on Android?

Android supports communication between two mobile applications via Intent (shared space between apps). So 3rd party software doesn't need to directly interface with the vendor SDK and hardware. Vendor application decodes the scanned barcode and puts it on Intent with a shared key. MAF Application reads the barcode information from Intent via a shared key. Unlike Android, iOS doesn't support communication between two mobile apps. So we need to directly interface with a specific vendor SDK. (No separate app from vendor). The Vendor SDK needs to decode scanned barcode and pass it to the Cordova plugin which then communicates this information to MAF application.

Drawbacks:

- All Vendor SDKs need be bundled while building MAF app.
- Cordova plugin has to be native (Objective-C) to interact with vendor SDK

Options available for iOS

Option A: Use wedge scanner (Hardware solution w/ sticky focus flag enabled)

Use wedge scanner (Hardware solution w/SIM's sticky focus flag enabled)

For iOS devices, there is no built-in scanner support. In order to achieve scanning, a wedge-scanner, either wired or Bluetooth enabled, can be used. SOCS has a system configuration option to enable '*Scan Focus Item Detail*' (focus stays on same field), so when a user selects an entry field all scans done by the wedge-scanner get captured for that field.

Sticky Focus flag is set here > `.adf\META-INF\adf-config.xml`

Option B: Use camera barcode scanner

- Go to the MAF Sample Applications directory for their BarcodeDemo project (/jdev/extensions/oracle.maf/Samples/PublicSamples/BarcodeDemo/src) and copy the folder phonegap-plugin-barcodescanner-master to your SIM Mobile MAA Project dir/plugins.
- Rename the folder to phonegap-plugin-barcodescanner in your SIM Mobile plugin folder.
- In your SOCS Mobile MAA project in JDeveloper, go to Application Resources > Descriptors > *ADF META-INF* and open *maf-plugins.xml*.
- Add the following cordova plugin entry:
- Go into *adf-config.xml* and set *DEVICE_CAMERA_SCAN* to true.
- Rebuild and redeploy the application.

Any device with a MAF-usable camera should now have a camera scan option on any Scan Bar inside the SOCS Mobile app.

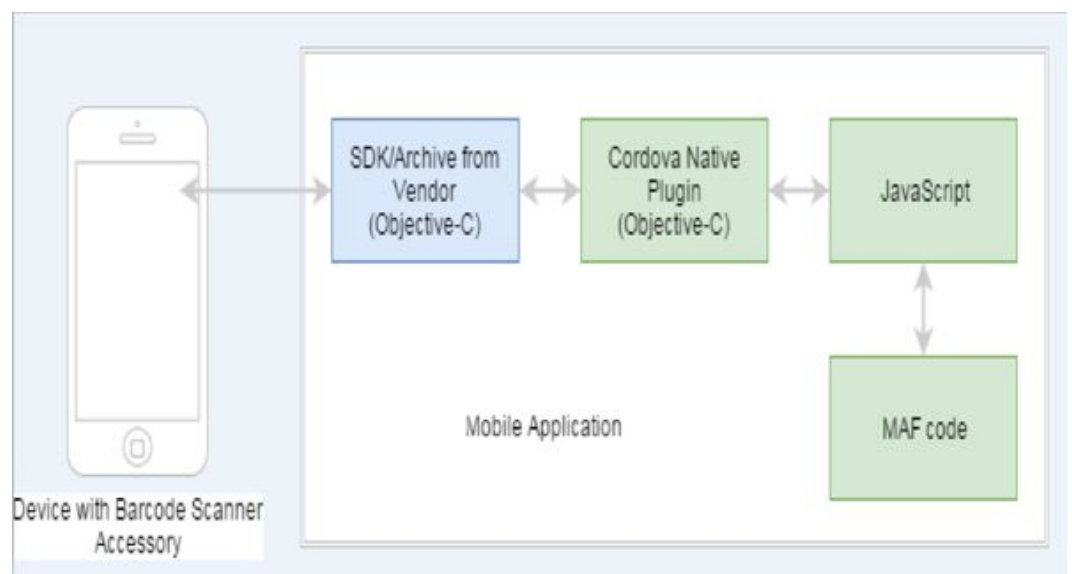
Camera scanning is not really a fast-scan (scan-scan) option, but will work in for small fashion store with low-volume of scanning.

Downside for any camera based scans can be dependent on focus & ambient lighting situation.

Option C: Customize code with specific vendor sled using objective-C

1. Archive SDK from a vendor (VeriFone, Honeywell, Infinite peripherals etc.) that communicates with the hardware barcode API's into the project. All the barcode decoding/encoding happens at this layer. This SDK would be in Objective – C.

2. Write a native Cordova plug-in in Objective-C which acts as an interface between SIM and Vendor SDK. More on iOS Cordova plug-in can be found at <https://cordova.apache.org/docs/en/latest/guide/platforms/ios/plugin.html>.



Archive SDK from a vendor (VeriFone, Honeywell, Infinite peripherals etc.) that communicates with the hardware barcode API's into the project.

All the barcode decoding/encoding happens at this layer. This SDK would be in Objective – C.

Write a native Cordova plug-in in Objective-C which acts as an interface between MAF mobile app and Vendor SDK.

More on iOS Cordova plug-in can be found at

<https://cordova.apache.org/docs/en/latest/guide/platforms/ios/plugin.html>.

How to develop a custom plugin?

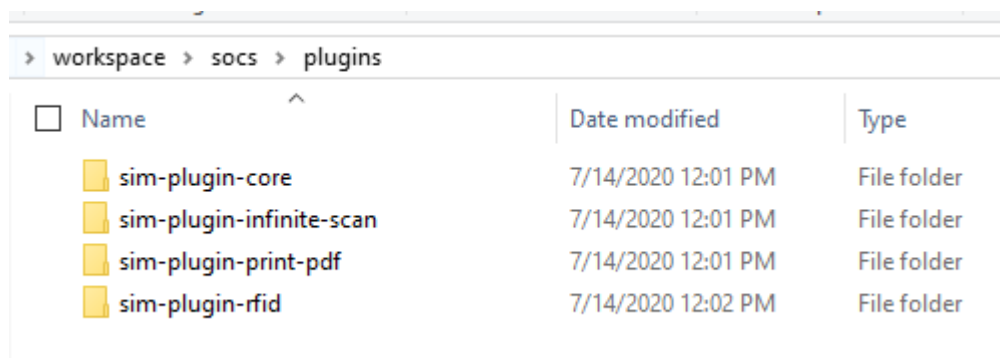
Please read the [Plugin Development Guide](#) for an overview of the plugin's structure and its common JavaScript interface. In addition to that also read the [iOS Plugin Development Guide](#) which provides details on how to implement the native plugin code on the iOS platform.

Finally, read the 'Introduction to custom Cordova Plugin Development'

at <https://blogs.oracle.com/digitalassistant/post/introduction-to-custom-cordova-plugin-development> for information on creating a very simple plugin for Android and iOS. These documentation resources will help you develop and build your own plugin if needed.

How to register plugins?

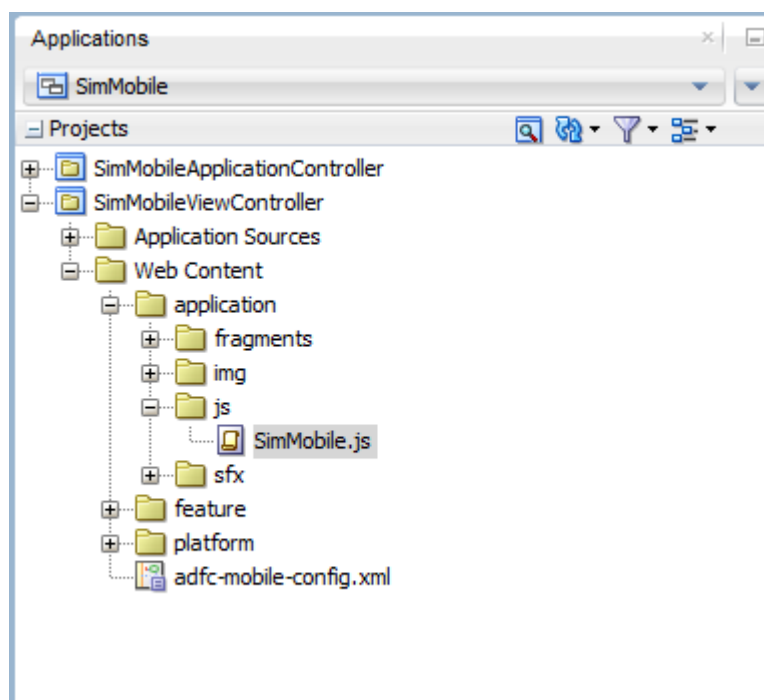
Implementers need to register the additional scanner plugins in to the SOCS application. Review [How to Register an Additional Plugin](#) for MAF application for the steps. Before you begin, ensure that the SOCS application and the plugin to be registered with the application, are stored on the same drive. Or you can place the plugin in SOCS workspace/plugins directory with the other SOCS provided plugins.



Name	Date modified	Type
sim-plugin-core	7/14/2020 12:01 PM	File folder
sim-plugin-infinite-scan	7/14/2020 12:01 PM	File folder
sim-plugin-print-pdf	7/14/2020 12:01 PM	File folder
sim-plugin-rfid	7/14/2020 12:02 PM	File folder

How to integrate custom Cordova plugin into MAF application?

After registering the custom scanner plugin, the plugin needs to be invoked to use the barcode scanner application on the device. Read the 'Integrating a custom Cordova plugin into a MAF app' at https://blogs.oracle.com/digitalassistant/post/entry/integrating_a_custom_cordova_plugin for information about how you can invoke a plugin from Java, from a MAF AMX page, and from local HTML. In SOCS, the plugin methods are invoked in *SimMobile.js*, located as shown in below:



Code snippet from an existing RFID plugin invoking registerScanner method:

```
retail.sim.initializeSimRfidPlugin = function () {
    adf.mf.log.Framework.logp(adf.mf.log.level.FINE, "SimMobile.js",
    "initializeSimRfidPlugin", "Initializing SIM RFID plugin.");

    try {
        cordova.plugins.retail.sim.rfid.registerScanner(retail.sim._scanBarcodeSuccess,
        retail.sim._scanBarcodeError);
    }

    catch (error) {
```

```
adf.mf.log.Framework.logp(adf.mf.log.level.SEVERE, "SimMobile.js",
"initializeSimRfidPlugin", "Error initializing SIM RFID plugin: " + error);

}}
```

Options available for Universal Windows Platform

Oracle MAF technology now supports desktop and tablet on Windows 10, but not mobile phones. There is no built-in scanner support for the Windows 10 tablet.

Windows Scanning options

Use wedge-scanner (either Bluetooth or wired – with sticky focus flag enabled)

Wedge scanners are generic scanner solutions that can be used without writing any custom code. They can be either Bluetooth-enabled or wired. The only caveat is they need to have focus enabled in a field in order to scan. SOCS has a system configuration option to *enable 'Scan Focus Item Detail'* (focus stays on same field), so when a user selects an entry field all scans done by the scanner/wedge-scanner get captured for that field.

Sticky Focus flag can be configured here > .adf\META-INF\adf-config.xml

Supported plugins

Vendor	Plugin	OS	Purpose	Comments
Zebra	sim.mobile.plugin.core	Android	Barcode Scanning	Used with DataWedge
VeriFone	sim.mobile.plugin.core	iOS	Barcode Scanning	Used with Veri
Infinite Peripherals	sim.mobile.plugin.infinite.scan	iOS	Barcode Scanning	Used with Infi
Zebra	sim.mobile.plugin.rfid	Android iOS	RFID Scanning	Used with Zeb

To Use these plugins:

1. Make sure the Bluetooth scanning device is paired with the mobile device.
2. Go into SOCS app and Navigate to Configuration > External Scanners. You should see a list of Bluetooth scanners attached to the device. Now select the one you want to connect to.
3. You will get a confirmation prompt, if the device is successfully connected.
4. SOCS should now be connected to the scanner.

Reconfiguring Integrated Scanner Plugin for Android

This section gives information about where and how base sim-plugin-core Cordova plugin would need to be modified, to allow for integration of Android Intents from non-DataWedge software. Please note that since intent handling differs with Android OS version & mobile device make/model, it is not possible to genericize intent handling. So please refer to how that specific device handles android-intent.

Details:

The Zebra DataWedge integrated device scanner communicates with device hardware via a Cordova plugin.

The source for this plugin can be accessed in the JDeveloper project directory of Oracle SOCS imported MAA or Mobile Application Archive.

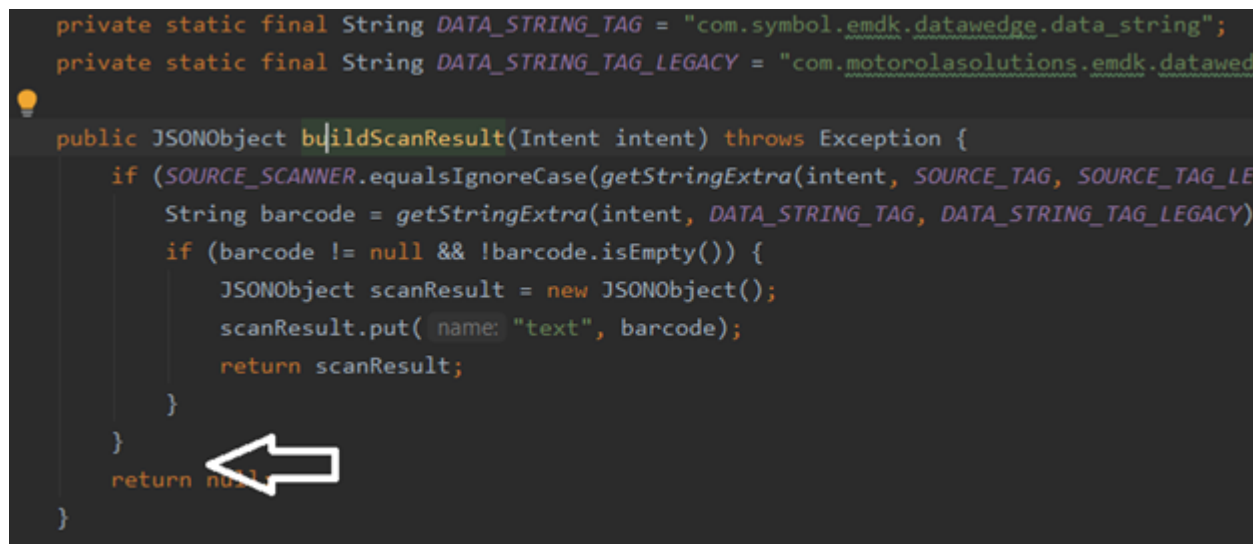
A sample directory being:

```
<JDeveloper Project Dir>/plugins/sim -plugin-  
core/src/android/oracle/retail/sim/mobile/plugin/core/
```

Within this directory, there would be a number of .java files. In order to change how Oracle SOCS decodes an Android Intent message from integrated scanning software, an implementer would need to edit the following file:

ZebraDataWedgeProvider.java

Within this file you will need to add code in following location:



```
private static final String DATA_STRING_TAG = "com.symbol.emdk.datawedge.data_string";  
private static final String DATA_STRING_TAG_LEGACY = "com.motorolasolutions.emdk.datawed  
  
public JSONObject buildScanResult(Intent intent) throws Exception {  
    if (SOURCE_SCANNER.equalsIgnoreCase(getStringExtra(intent, SOURCE_TAG, SOURCE_TAG_LE  
        String barcode = getStringExtra(intent, DATA_STRING_TAG, DATA_STRING_TAG_LEGACY)  
        if (barcode != null && !barcode.isEmpty()) {  
            JSONObject scanResult = new JSONObject();  
            scanResult.put( name: "text", barcode);  
            return scanResult;  
        }  
    }  
    return null;  
}
```

Here you will need to insert your own code to detect and un-package contents of the Android Intent that is sent from the specific integrated scanner used on your device.

Oracle SOCS will need the single string of the complete barcode, and this barcode will need to be packaged into a JSONObject just as the above snippet of code does, into an object like:

```
{  
    text : <barcode contents>  
}
```

Once this JSONObject is returned from the above method, the rest of the Oracle SOCS code should be able to function correctly.

Didn't find what you are looking for?