Integrating DataWedge into your Cordova application

Posted by Darryn Campbell in Android Blogs on Aug 4, 2016 1:31:24 AM

This blog accompanies an application hosted in github at GitHub - darryncampbell/DataWedgeCordova: Cordova application using DataWedge solely through Intents and shows how DataWedge functionality can be seamlessly integrated into your new or existing Cordova / Ionic / Phonegap applications using 3rd party plugins

DataWedge Intent Interface

DataWedge is a value-add of all Zebra Technologies devices (formally Symbol and Motorola Solutions) that allows barcode capture and configuration without the need to write any code. This application will demonstrate how to use Android intents to add DataWedge scanning functionality to your application

Quick Start - Getting this Demo running

1. >git clone https://github.com/darryncampbell/DataWedgeCordova.git
2. >cd DataWedgeCordova
3. >cordova platform add android
4. Plug in Zebra device
5. Follow instructions under "Configuring Datawedge" (below)
6. >cordova run android
7. Scan a barcode

Getting Started

This section walks through the steps to create a new Cordova application that utilises DataWedge for scanning.

Create a cordova application that will run on Android

1. cordova create DataWedgeCordova com.zebra.datawedgecordova DataWedgeCordova
2. cordova platform android

We will use a couple of 3rd party plugin to handle sending and receiving Intents to the DataWedge service. Any plugins capable of sending or receiving generic intents and interpreting the extra bundle into JSON will work:

For receiving intents:

1. cordova plugin add https://github.com/napolitano/cordova-plugin-intent#v0.1.31

For sending intents:

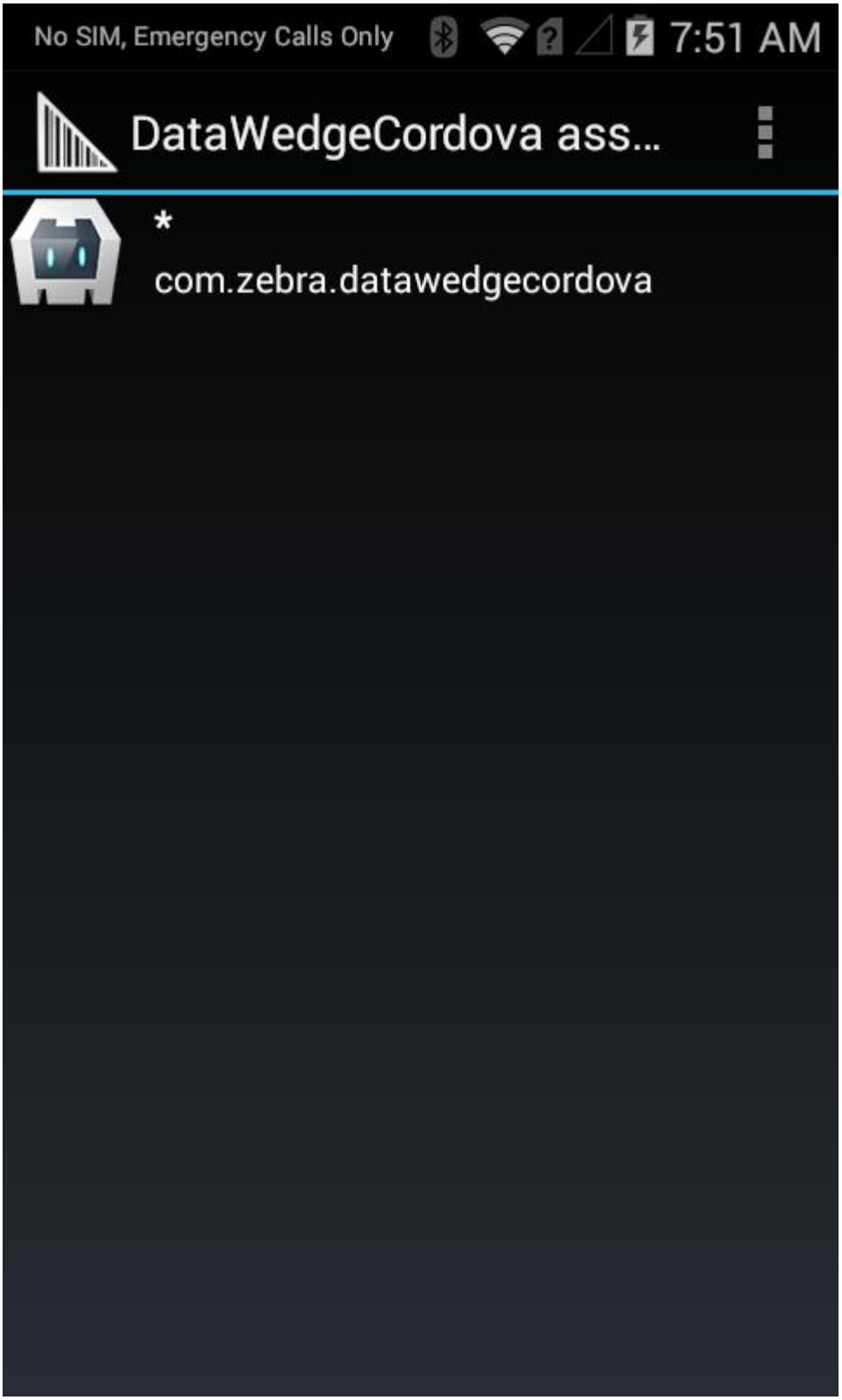
1. cordova plugin add https://github.com/Initsogar/cordova-webintent.git

The author of the plugin we'll use to receive intents very helpfully added parsing of intent extras in the 1.31 release so we'll explicitly grab that version.

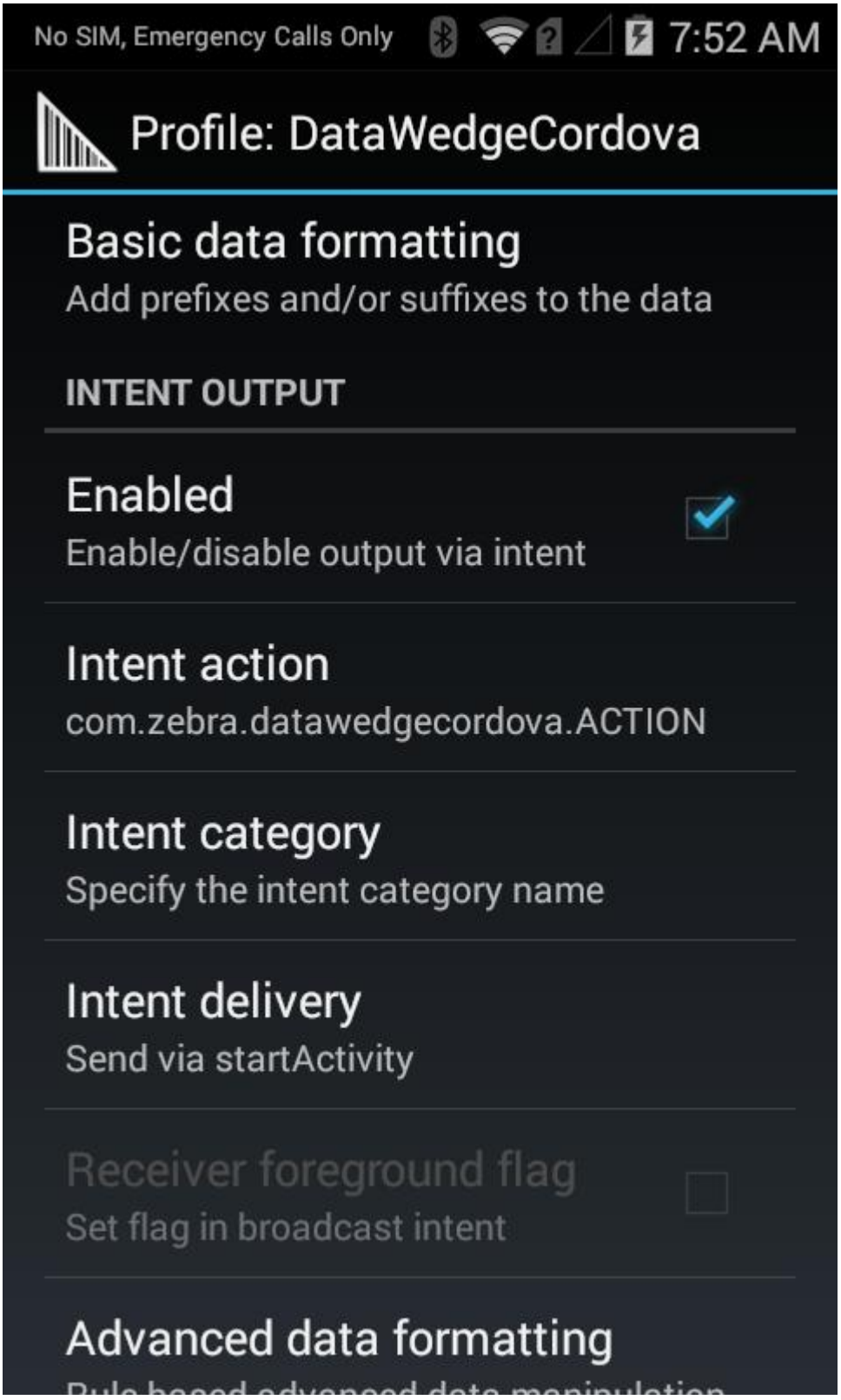
Configuring Datawedge

DataWedge can be configured to send intents whenever it scans barcodes. More detailed help is available at the official documentation but for the purposes of this demo I will include the pertinent steps. There are more sophisticated ways to configure DataWedge but this section will cover the basics.

- Create a new DataWedge profile (Applications --> DataWedge --> Menu --> New Profile). This will be the profile that will be active when our Cordova application is in the foreground. Give it a name e.g. DataWedgeCordova and click into it to configure.
- The next step requires our Cordova application to have previously run on the device so if you have not already done so, `cordova run android`
- Back in DataWedge configuration associate the Cordova application with our DW profile



- Scroll down to the Intent Output section of DW configuration and enable intents to start our activity:
  - Intent Action: com.zebra.datawedgecordova.ACTION
  - Intent Category: leave blank
  - Intent delivery: Start Activity



Add Intent Filter to the Cordova Application

By default our 3rd party plugin handling intents will not know to listen for intents carrying com.zebra.datawedgecordova.ACTION. There are multiple ways to configure our Cordova app to listen for this intent, we could create a custom plugin whose config.xml contains the intent-filter or we could get

more complicated and dynamically register a broadcast listener in our plugin (assuming the DW intent delivery is modified to broadcast of course).

The simplest technique for the purposes of this demonstration is to just manually insert the listener into our AndroidManifest.xml. This is not ideal as the manifest will be overwritten if we remove and re-add our Android platform but for most purposes this is fine. To make this demo more reliable I have added a build hook to copy a predefined AndroidManifest.xml during the build process: DataWedgeCordova\platforms\android\AndroidManifest.xml:

```
1.      <activity ... android:launchMode="singleTop" ... >
2.        <intent-filter android:label="@string/launcher_name">
3.        ...
4.        </intent-filter>
5.  <intent-filter>
6.            <action android:name="com.zebra.datawedgecordova.ACTION" />
7.            <category android:name="android.intent.category.DEFAULT" />
8.        </intent-filter>`
9.      </activity>
```

Add Code to the Cordova application

Now to hook up our logic to listen for and send intents.

Listening for intents

For sake of brevity modify the onDeviceReady function to call out to our 3rd party intent plugin and register a handler for new Intents. Assuming this is a DataWedge intent, process the data and display the barcode on the screen:

```
1.    onDeviceReady: function() {
2.        app.receivedEvent('deviceready');
3.
4.        window.plugins.intent.setNewIntentHandler(function (intent) {
5.            console.log('Received Intent: ' + JSON.stringify(intent.extras));
6.            var decodedBarcode = intent.extras["com.symbol.datawedge.data_string"];
7.            var parentElement = document.getElementById('barcodeData');
8.            if (parentElement && decodedBarcode)
9.            {
10.               parentElement.innerHTML = "Barcode: " + decodedBarcode;
11.               parentElement.setAttribute('style', 'background-color:#0077A0;color:#FFFFFF;');
12.           }
13.       });
14.   },
```

And in index.html:

```
1.  <div class="event" id="barcodeData"></div>
```

Sending intents

DataWedge supports an intent based API documented here. The API is somewhat limited but supports initiating the scanner via software (simulating a trigger press), disabling the scanner entirely and various

DataWedge profile operations. We'll be adding features to our application to simulate a trigger press and disable / enable the scanner through that Intent interface:

To simulate a trigger press:

```
window.plugins.webintent.sendBroadcast({
    action: 'com.symbol.datawedge.api.ACTION_SOFTSCANTRIGGER',
    extras: {
        'com.symbol.datawedge.api.EXTRA_PARAMETER': 'START_SCANNING'
        }
    },
    function() {},
    function() {}
);
```

To disable the scanner:

```
window.plugins.webintent.sendBroadcast({
    action: 'com.symbol.datawedge.api.ACTION_SCANNERINPUTPLUGIN',
    extras: {
        'com.symbol.datawedge.api.EXTRA_PARAMETER': 'DISABLE_PLUGIN'
        }
    },
    function() {},
    function() {}
);
```

And hook up our logic to the UI:

```
document.getElementById("scanButton").addEventListener("click", startSoftTrigger);
document.getElementById("disableScanningButton").addEventListener("click", disableEnableScanning);
```

Now deploy & launch your app. You are able to scan barcodes and exercise the functionality:

Notes

The activity launchMode needs to be set to 'singleTop' in Cordova, this can be achieved by setting `<preference name="AndroidLaunchMode" value="singleTop" />` though in my experience the lauchMode is singleTop by default. This will ensure that each scan does not launch a new instance of the application.