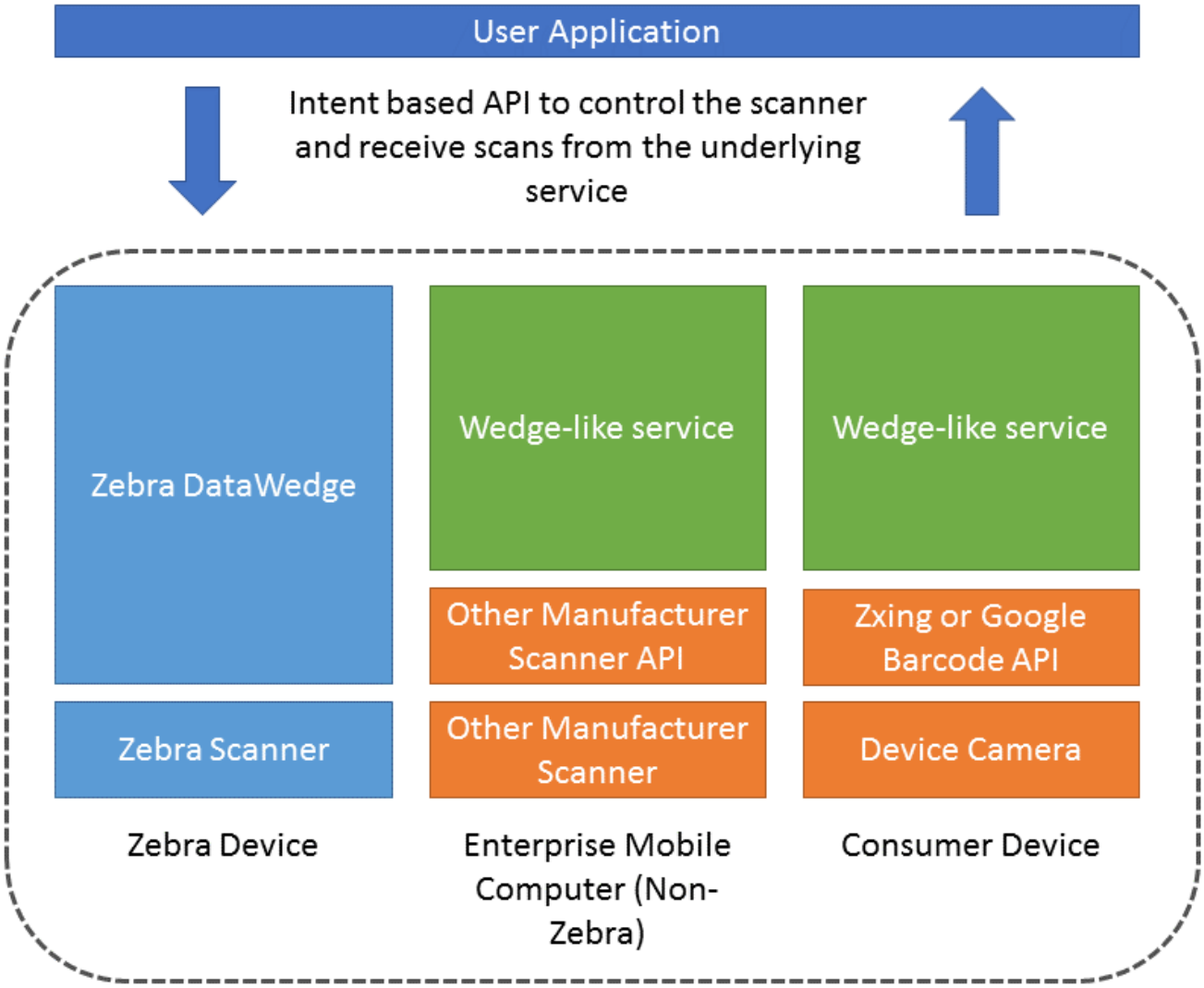
DARRYN CAMPBELL
Mobile computing and enterprise software development




# Writing Enterprise Android applications that capture barcode data and run on multiple devices

*16th August 2016* *0* *By* DARRYNCAMPBELL

Capturing barcodes in a consumer application is easy, right? Just importing the ZXing library into your application gives your user the ability to capture the occasional barcode. Users can then now a price

look-up or maybe scan the odd QR code. If you're writing a cordova application then it's arguably even easier: just add one of the plethora of Barcode plugins to your app (e.g. https://github.com/wildabeast/BarcodeScanner) and you're away. Similarly writing a Xamarin application there are multiple scanning SDK components for you to easily integrate with your C# application. The consumer angle is covered and in the case of Cordova and Xamarin that same application could also capture barcodes on an iOS device with minimal modifications – brilliant for cross-device compatibility!

Now consider Enterprise barcode capture, mobile computers from companies like Zebra, Honeywell and DataLogic integrate dedicated barcode capture hardware making it much quicker and infinitely more ergonomic to capture barcodes on a ruggedized device without having to resort to the device's camera hardware:


The Honeywell CT50, the Datalogic Axist and the Zebra TC8000.

All of the above devices run Google's Android operating system with varying levels of enhancements to meet the unique requirements of Enterprises such as device management, app management and ruggedization.

Without fail your application will need to call a manufacturer specific SDK in order to control and modify the scanning hardware. All manufacturers offer a native Android SDK but there is not consistent support for Xamarin or Javascript. It also goes without saying that the APIs offered by each manufacturer are different.

As an application developer or ISV you want to write applications that run across as many platforms as possible without having to write a different application for each manufacturer's devices. Your clients are also showing a strong interest in consumer devices so you would like some way to have your application work on those consumer devices and do barcode scanning through ZXing, Google's Vision Barcode API or a Bluetooth connected scanning sled.

You might also have an application that ran perfectly fine on Windows Mobile but as clients move to Android based devices you are faced with the challenge of porting those applications to a new Operating system.

One obvious solution would be to abstract the manufacturer specific APIs into some uber-set of functionality so your new class dynamically detects whether it is running on a Zebra device, a Honeywell device or any other type of supported device and dynamically calls the appropriate SDK. This is probably technically possible (I've never tried it!) but would require including many manufacturer specific libraries

into in every application and any change to the underlying libraries could necessitate a re-validation of every dependent application.

Another solution would be to leverage the wedge solution offered by Zebra.

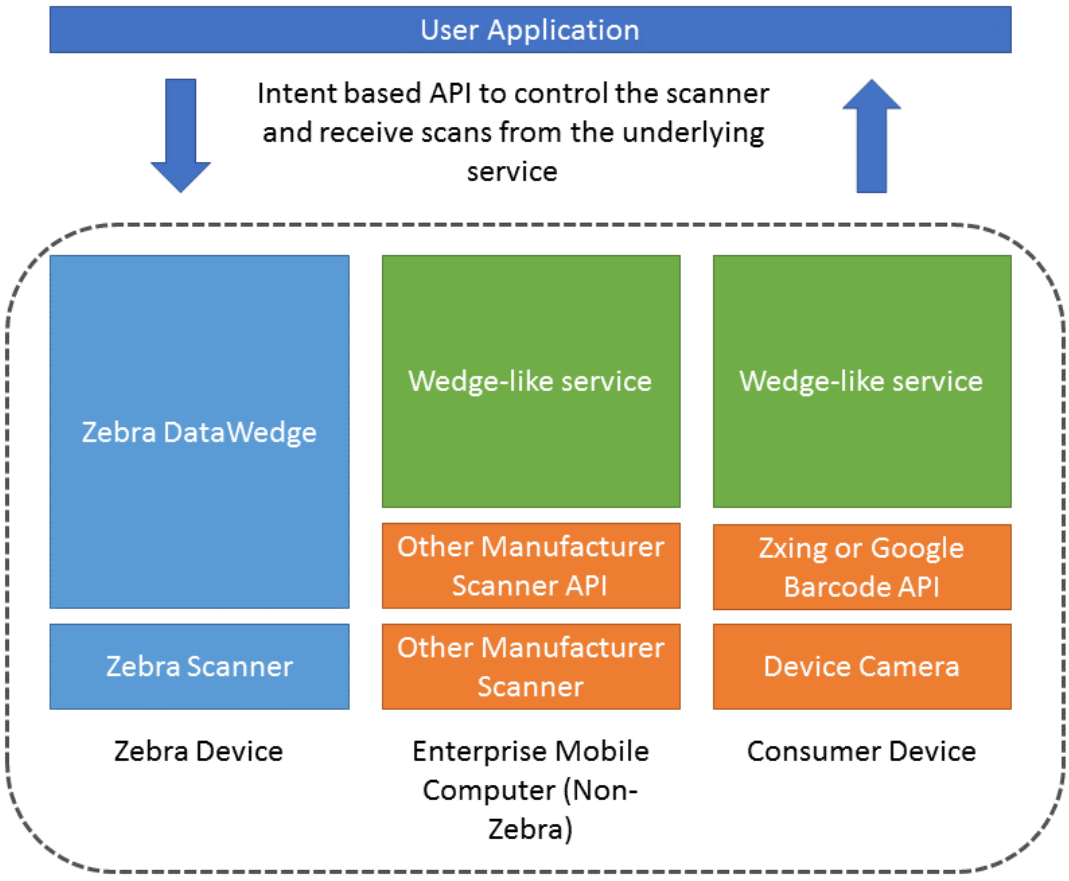
# DataWedge

In years gone by every manufacturer offered a comparable 'wedge' solution, that is a service running on the device which would control the barcode scanner and when the user scanned a barcode the foreground application would receive the decoded data as a series of keystrokes. Doing so your application loses some control over the scanner but you gain cross-device compatibility, since the application is receiving scans as keystrokes and there is no API the application does not care about the underlying hardware.

DataWedge is still offered on Zebra devices and now offers greatly improved functionality over those early 'wedge' solutions. DataWedge has full Android support, allows full control over the barcode scanner, integrates with solutions for OCR, OMR (optical mark recognition) and can return scanned data via Android intents.

An application that makes use of DataWedge does not have to consider the underlying hardware, it just receives intents when barcodes are scanned and has some rudimentary configuration over Datawedge through an Intent API

It would be good if that same application that uses DataWedge on Zebra devices can be run on a different manufacturer's device or any Android consumer device and 'just work', using the Intent API to talk to some service on the device that mimics DataWedge. The service would be responsible for calling the appropriate manufacturer SDK or use a consumer focused barcode API:




I have implemented the 'Wedge-like service' (in green in the diagram) at https://github.com/darryncampbell/GenericScanWedge . Currently however the service only supports

ZXing and the Google Barcode API and does not support enterprise mobile computers. See the readme for instructions on how to use the service and to make it easier to test it you can use the "Datawedge API Exerciser" which is a bare-bones application that makes use of the DataWedge API.

**Share this:**

**Related**

Instrumented Testing and the Zebra EMDK Barcode API
1st March 2017
In "Zebra Technologies"

DataWedge to WebSockets Bridge
11th June 2017
In "Zebra Technologies"

AppForums 2019: NALA (Las Vegas)
27th August 2019
In "Speaking"

Category Zebra Technologies

Tags Android DataWedge

## Leave a Reply

Your email address will not be published. Required fields are marked *

**Comment**

**Name ***

**Email ***

**Website**

**Notify me of follow-up comments by email.**