# Kotlin and developing Kotlin applications for Zebra devices

*Submitted by Anonymous (not verified) on November 19, 2018 - 6:01am*

**Tags:** Emdk (/taxonomy/term/185) android (/taxonomy/term/141) datawedge api (/taxonomy/term/170) emdk for android (/taxonomy/term/183) jive-migrated (/taxonomy/term/370)

It is not news for Android developers that following Google's announcement of Kotlin support (https://android-developers.googleblog.com/2017/05/android-announces-support-for-kotlin.html) during Google I/O 2017, Google have embraced Kotlin as the preferred development language for Android, even to the extent that the "Build your first app (https://developer.android.com/training/basics/firstapp/starting-activity)" tutorial on the Android developer site will default to showing Kotlin code.

The Stack Overflow 2018 developer survey (https://insights.stackoverflow.com/survey/2018/#overview) gives a non-biased evaluation of the current mindshare of Kotlin: 4.7% of professional developers listed Kotlin as the most popular technology (https://insights.stackoverflow.com/survey/2018/#most-popular-technologies) which might not sound high, but it came second in the most loved (https://insights.stackoverflow.com/survey/2018/#most-loved-dreaded-and-wanted) and fourth in the most wanted (https://insights.stackoverflow.com/survey/2018/#most-loved-dreaded-and-wanted) categories. Developers who listed Kotlin had the fewest years of professional coding experience (https://insights.stackoverflow.com/survey/2018/#experience) which is understandable for a language which has only recently seen widespread adoption (this is the first year Stack Overflow have asked about Kotlin).

I was fortunate enough to attend the European Kotlin Conference (https://kotlinconf.com/) last month and I spoke with many developers and sponsors about using the language for enterprise development. There is adoption of Kotlin in the enterprise, although the majority of enterprise Kotlin developers right now are only targeting BYOD use cases, this is very likely to change in the near future.

Given:

- Google's continued promotion of Kotlin over Java since I/O 2017
- An awareness of Zebra and ruggedized devices amongst the Kotlin developers I spoke with at KotlinConf
- The popularity of the language amongst developers in general

I felt it made sense to provide some guidance around how to develop a Kotlin app targeting Zebra devices. There are 2 approaches you could take:

## Approach #1: Calling the EMDK library from Kotlin

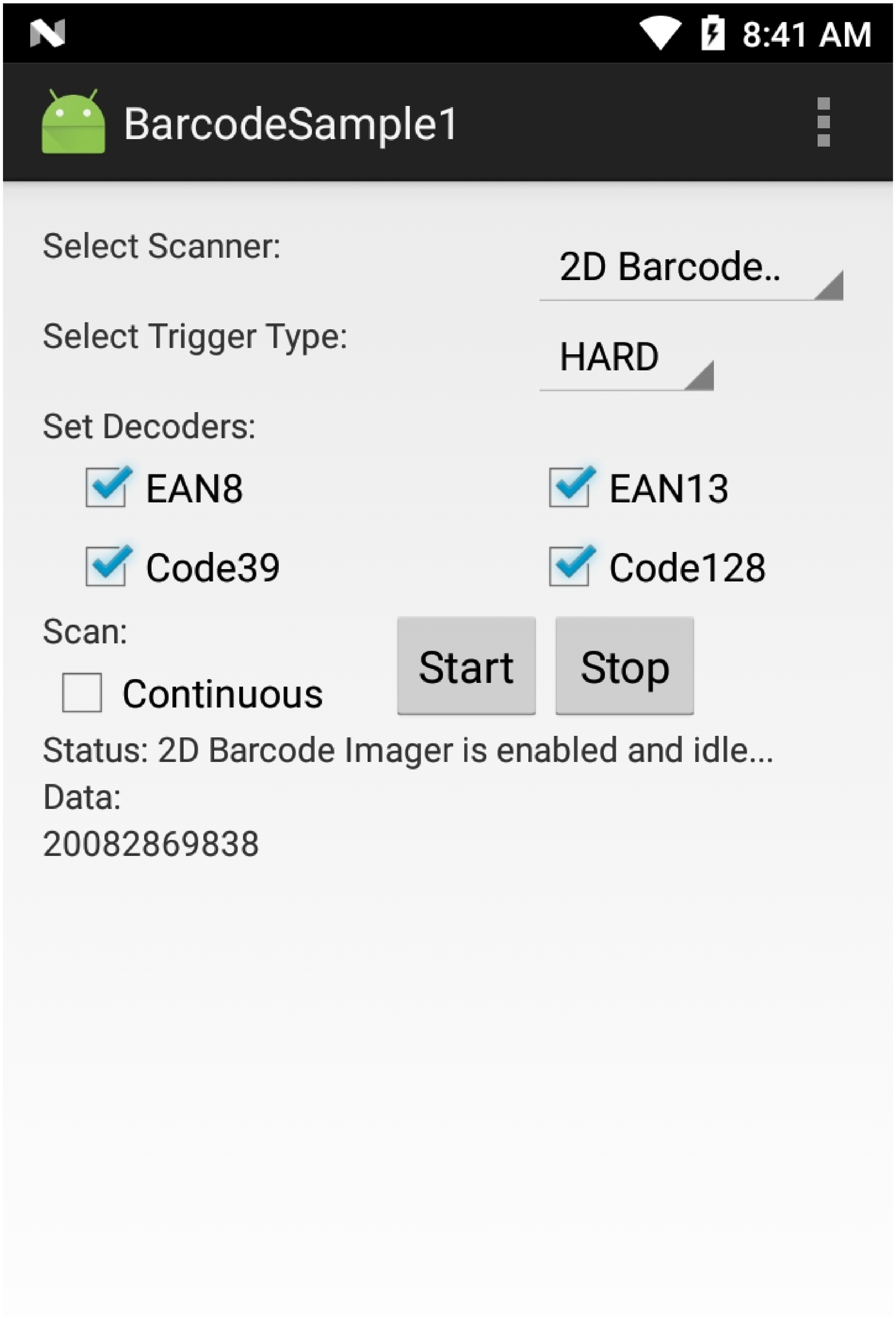
Java and Kotlin are completely interoperable. It is entirely possible to write a Kotlin application to consume Zebra's EMDK for Android (http://techdocs.zebra.com/emdk-for-android/latest/guide/about/) library and I have created a proof of concept in this forked branch (https://github.com/darryncampbell/samples-emdkforandroid-6_9/tree/BarcodeSample1-Kotlin/BarcodeSample1) based on the standard EMDK Barcode Sample 1 (http://techdocs.zebra.com/emdk-for-android/6-9/samples/barcode/) (the important file is MainActivity.kt (https://github.com/darryncampbell/samples-emdkforandroid-6_9/blob/BarcodeSample1-Kotlin/BarcodeSample1/app/src/main/java/com/symbol/barcodesample1/MainActivity.kt)). This fork (https://github.com/darryncampbell/samples-emdkforandroid-6_9/tree/BarcodeSample1-Kotlin/BarcodeSample1) can be built with Android Studio 3.x and run on any Zebra mobile computer supported by EMDK 6.9 (http://techdocs.zebra.com/emdk-for-android/6-9/guide/about/).

The proof of concept does not do anything particularly clever, the steps I took were as follows:

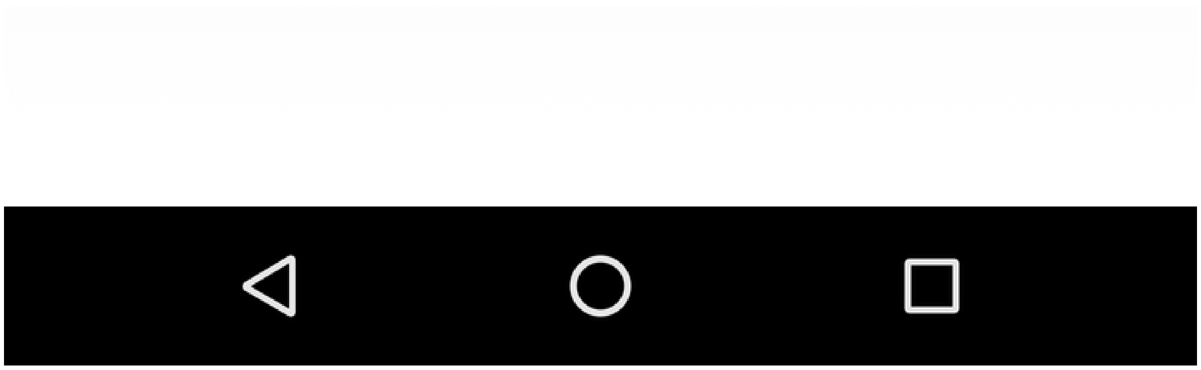
1. Fork the official EMDK for Android Barcode Sample 1 (I could have chosen any version of EMDK, I just chose 6.9 as it was the most recent at the time).
2. Use the Android Studio feature 'Convert Java file to Kotlin file', CTRL+ALT+SHIFT+K
3. Tidied up the very few conversion issues, raised by the compiler.

You might get a better view of the differences between the fork and the master branch by comparing a pull request (https://github.com/darryncampbell/samples-emdkforandroid-6_9/compare/master...darryncampbell:BarcodeSample1-Kotlin?expand=1) or viewing the Kotlin file directly (https://github.com/darryncampbell/samples-emdkforandroid-6_9/blob/BarcodeSample1-Kotlin/BarcodeSample1/app/src/main/java/com/symbol/barcodesample1/MainActivity.kt).

There are some changes the implementation team are working on to make the EMDK library more friendly to Kotlin developers but there is nothing stopping a Kotlin developer using the EMDK library today.



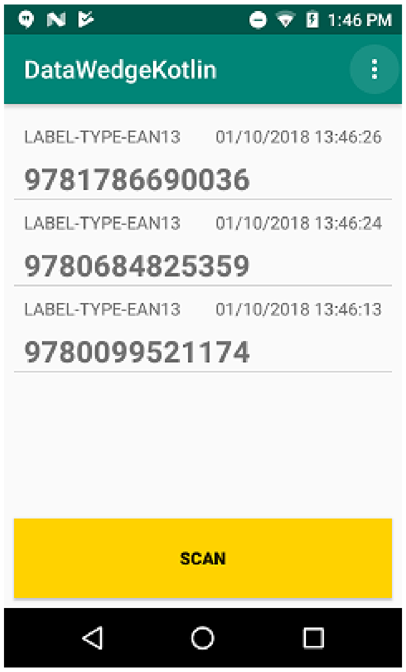
Status: 2D Barcode Imager is enabled and idle...
Data:
20082869838



## Approach #2: Using the DataWedge Intent API from Kotlin

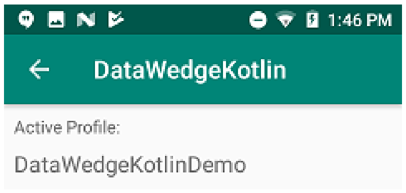
DataWedge exposes an Intent-based API. Many customers use DataWedge in preference to our native SDKs because it takes away all concerns about interfacing with the scanning hardware and reduces the task to integrating with a pre-built service. I have previously written about using the DataWedge API for Android (Java) apps (http://techdocs.zebra.com/datawedge/6-8/guide/api/tutorials/), Xamarin apps (/community/home/blog/2018/07/24/tutorial-scanning-with-datawedge-and-xamarin), Ionic apps (/community/home/blog/2018/04/03/ionic-applications-on-zebra-devices), Cordova apps (/community/home/blog/2016/08/04/integrating-datawedge-into-your-cordova-application) and React Native apps (/community/home/blog/2018/10/29/developing-react-native-applications-on-zebra-devices)… it makes sense to continue the pattern and show how a Kotlin application could be developed to take advantage of the DataWedge APIs.

My DataWedgeKotlin (https://github.com/darryncampbell/DataWedgeKotlin) application is available from github at https://github.com/darryncampbell/DataWedgeKotlin (https://github.com/darryncampbell/DataWedgeKotlin), please clone the repository and follow the instruction in the ReadMe file (https://github.com/darryncampbell/DataWedgeKotlin/blob/master/README.md) to get started. The application comprises two activities, the first (https://github.com/darryncampbell/DataWedgeKotlin/blob/master/app/src/main/java/com/darryncampbell/datawedgekotlin/MainActivity.kt) displays received scans and can invoke the scanner with a soft button.

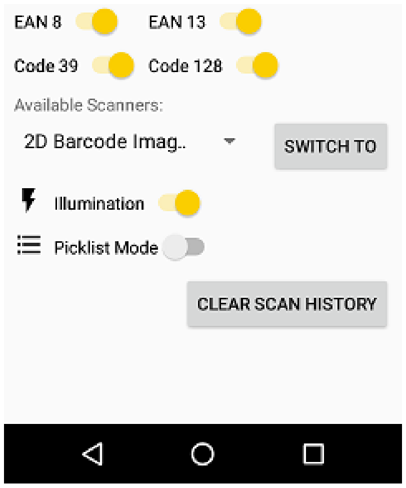


LABEL-TYPE-EAN13 01/10/2018 13:46:26
9781786690036
LABEL-TYPE-EAN13 01/10/2018 13:46:24
9780684825359
LABEL-TYPE-EAN13 01/10/2018 13:46:13
9780099521174

The second activity (https://github.com/darryncampbell/DataWedgeKotlin/blob/master/app/src/main/java/com/darryncampbell/datawedgekotlin/ConfigurationActivity.kt) allows for configuration of the selected scanner and uses the DataWedge SET_CONFIG (http://techdocs.zebra.com/datawedge/latest/guide/api/setconfig/) API to make changes to the current scanner configuration.



Active Profile:
DataWedgeKotlinDemo



I think the source code (https://github.com/darryncampbell/DataWedgeKotlin/tree/master/app/src/main/java/com/darryncampbell/datawedgekotlin) should be fairly self-explanatory but I apologise if it is not *idiomatic* Kotlin – it is difficult having used Java since my university days.

Whichever way you choose to develop a Kotlin application on Zebra enterprise devices, please do also provide us feedback (either by replying to this post or by posting questions to the developer portal) – the more developers we see using Kotlin, the more we will focus on providing official, formally supported samples and documentation for the technology which will lead to a better experience all round.

(https://www.linkedin.com/company/167024?trk=tyah) (https://www.twitter.com/zebratechnology)

(https://www.facebook.com/pages/Zebra-Technologies/107703715989073) (https://www.youtube.com/zebratechnologies#p/u)
(https://www.zebra.com/us/en/blog.html)

Help (https://developer.zebra.com/taxonomy/term/291) | Terms of Use (https://developer.zebra.com/terms_of_service)
| Privacy Policy (https://www.zebra.com/us/en/about-zebra/company-information/legal/privacy-statement.html)
| Legal (https://www.zebra.com/us/en/about-zebra/company-information/legal.html)