Android development returns to Eclipse

Andrew Bowley

December 2018

Introducing the Andworx Project

What is Andworx?

Andworx was born in April 2018 with the aim of allowing Android developers to use Eclipse as an alternative to Android Studio. The focus, as with the original Android Development Tools (ADT), is on building and debugging applications. Andworx is a long way from being ready for general use, but an example application is available to demonstrate critical functionality:

- Import the application project into the Eclipse workspace
- Build the application debug version
- Launch the application on either an attached device or emulator
- Debug the application against the project sources
- Export a release APK

The current Andworx API level is 27 (Android 8.1), which means a lot of work has been done to incorporate contemporary Android tool chains into the Eclipse framework. This includes, for example, using Desugar to transform Java 8 classes into a format suitable for Android packaging.

Andworx also retains many of the tools that came with the original ADT. So you can view logging messages with LogCat and processes with Device Monitor. There is also the Android SDK Manager for installing packages using a GUI and Android Device Monitor for setting up emulators.

Andworx Architecture

Architectural decisions made at the start of a project have long term consequences and may be the most important factor in the success, or otherwise, of the project. One decision has been to avoid the Android Studio architecture of layering an IDE (IntelliiJ) on top of a Gradle foundation. This is despite the fact it would make achieving compatibility with Android Studio easy. However, Gradle is very heavyweight and conflicts with Eclipse in many ways. Unfortunately, Android build tools buy into Gradle technology deeply and not having the Gradle runtime libraries available makes development challenging. Anyway, Andworx has worked around the absence of Gradle, sometimes inventively, and given the progress achieved so far, it is reasonable to assume there will be ultimate success.

So what changes have been made to the original ADT architecture to arrive at where Andworx is today?

1. There are 2 additional application layers, one for building projects on top of which is a workspace modelling layer for dealing with multi-module project structures
2. Android library dependencies are resolved by M2E Maven integration for Eclipse
3. AAR dependencies are expanded into a dedicated Maven repository
4. Source sets have been introduced equivalent to Gradle’s and with identical defaults
5. The staging of builds follow Gradle’s design including interrmediate artifact creation
6. Configuration stored using lightweight Java Persistence Architecture (JPA)

Project Status

Andworx is currently at proof-of-concept stage. Many important features are yet to be designed including:

- Create and configure new projects
- Testing support
- Supplemental rendering tools
- Incremental builds
- Creation of configuration variants and flavors
- Split APKs

The sources for Andworx are available publicly on Github at https://github.com/androidworx/andworx and there is an open invitation for developers to collaborate in moving the project forward. There are plenty of challenges for anyone tackling the outstanding features list. In addition, compatibility with Android Studio is a key issue in getting Andworx off the ground.

Android Studio Compatibility

An example application to demonstrate API 27 Runtime Permissions is established to showcase Andworx's capabilities. The application is built with support libraries to highlight behaviour when running on earlier Android versions. This project, as provided, is compatible with Android Studio, However, Andworx capability to resolve the Groovy scripting content of a build.gradle file is currently very limited. This Groovy issue needs to be dealt with before Andworx can have any workable Android Studio compatibility. However, 100% compatibility is likely not achievable given the issues around how to handle included Gradle plugins.

Another compatibility topic is the new Kotlin language currently supplanting Java for Android development. As Kotlin support is now available for Eclipse, is may be feasible to add Kotlin to the Andworx technology mix. However, the absence of Kotlin skills is preventing any initiative at this stage.

Andworx Install

Eclipse

Andworx is specifically targets Photon Eclipse IDE for Java Developers, It is recommended that you keep it updated to the latest version, which is 2018-12 at time of writing. Note that Java 8 is a prerequisite of both Eclipse Photon and Andworx. There are other perquisites if you intend to develop Andworx which you can find in the README.md.

Log4j

Eclipse needs to be configured for log4j, otherwise you will get an error message on the console at start up and also miss out on potentially useful logger information. ou can get a sample log4j configuration in XML format from Log4j XML Configuration Primer.

Install new software

The Andworx update site is Andworx December 2018 Release https://dl.bintray.com/cybersearch2/plugins/Andworx/V0.0.3. Install Andworx using this site in the normal manner. You see the site makes available both "Andworx" and "Andworx Core SDK". You need only select the latter if you intend to do Andworx development. The installation details shows a single Andworx feature "Development Tools for Android". After reviewing and accepting the license terms, the final step is to restart Eclipse.

Android SDK (API 27)

For more details on installing and updating the SDK, refer to SDK_Manager_Guide.pdf. After restarting Eclipse, you will need to configure the Android SDK to be used by Andworx. You will see a dialog pop up titled "Select Android SDK installation". If you do not see this dialog, it means the ADT or Andmore has been installed on you machine sometime in the past, in which case, you can use the SDK manager to update your SDK installation.

Hit the "Configure" button on the SDK dialog to either select an existing SDK or install a new one. If selecting an existing SDK, then uncheck the "Create new SDK" option. Either way, you need to ensure that the API 27 platform is installed as this is the Android level at which Andworx is designed to work.

APK Signing Configuration

A private security key is required to sign APKs that are to be installed on physical devices. Such a key is required for the Andworx demonstration described below. If you are unfamiliar with the Android Signing Configuration, take a look at Sign your app. The details of how to create a private key are system-specific, but you will need to store it in a keystore file and record the following details:

- keystore location
- keystore type (JKS, JCEKS or PKCS12)
- keystore password
- key password
- key alias

Runtime Permissions

Download and unzip the Runtime Permissions example application to somewhere convenient, but avoid the Eclipse workspace. The idea is to import a copy into the workspace so the original is always available for comparison or experimentation. The previous Andworx Install section covers all the perquisites for this application, including the installation of a private key to sign a release APK.

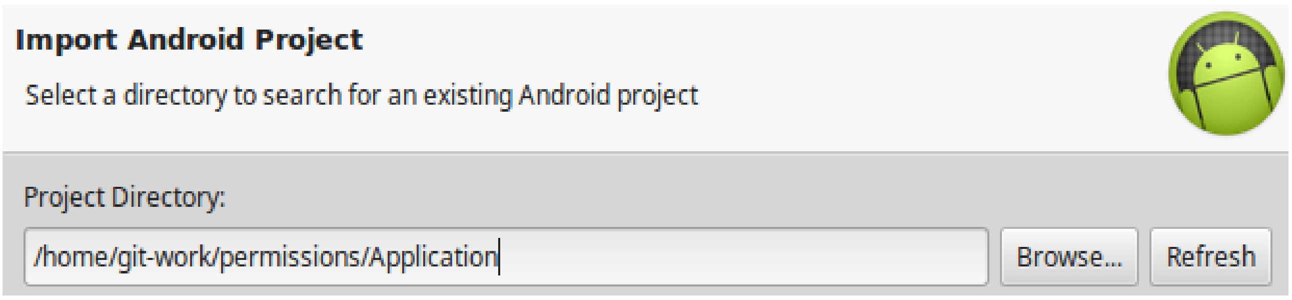
Take a look at Runtime Permissions, and you will see a typical example from the Android Developers Github site, but modified slightly to support creation of a release APK. There is both a Kotlin and Java version, but we will be looking only at the Java version in the folder under the root project named “Application”. In the root project folder, there is a “key.properties” file containing the values for the signing configuration. Edit this file to set the configuration to suit your development environment.

At this point you can familiarise yourself with the application. You can import the project into Android Studio or simply build and run the application from the command line.

Project Import

To import Runtime Permissions into Andworx, select from the top menu File —► Import —► Import Android Project —► Next.

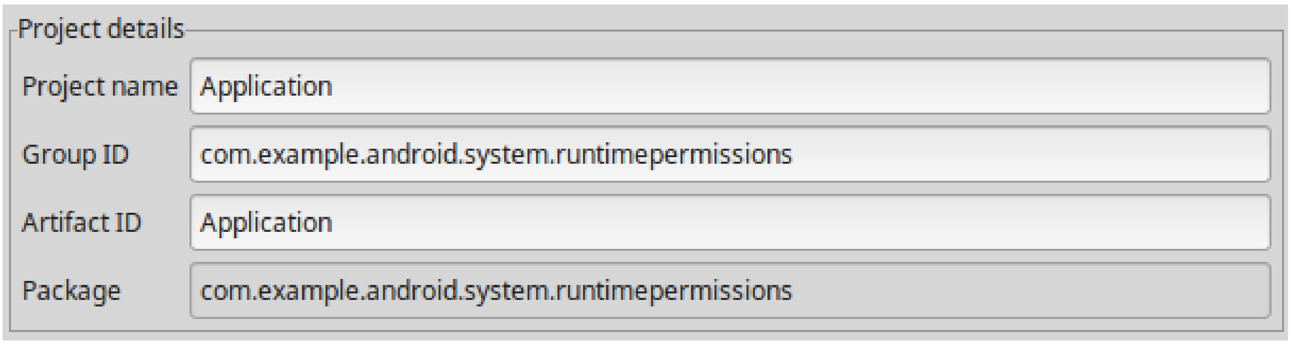
The Import Android Project dialog is explained by splitting into it’s components. At the top is the dialog title, prompt and project directory panel:


Import Android Project

Select a directory to search for an existing Android project

Project Directory: /home/git-work/permissions/Application

Select the project “Application” location where you unzipped Runtime Permission either by using the browse button or pasting it into the “Project Directory” field and hitting return. The next panel contains project details:



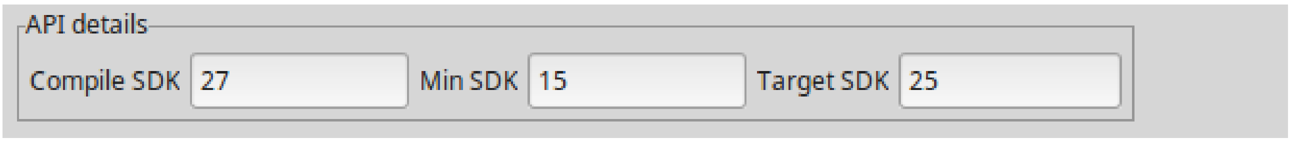
Project details

Project name	Application
Group ID	com.example.android.system.runtimepermissions
Artifact ID	Application
Package	com.example.android.system.runtimepermissions

You can see that the Andworx project profile includes repository coordinates Group id and Artifact id. The values shown come from the Android manifest package and project directory name. The project name default is also the same as for Artifact id. You should edit the project details to apply more suitable values, such as:

- Project name: com.example.android.Permissions
- Group ID: com.example.android
- Artifact ID: permissions

The next panel is API details:

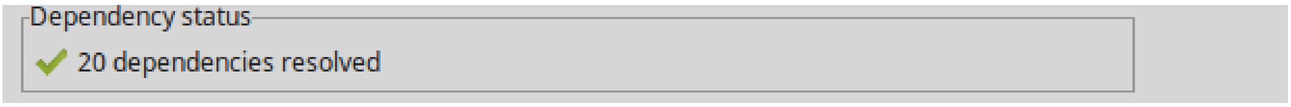


API details

Compile SDK	27	Min SDK	15	Target SDK	25

The API details are three parameters which define Android SDK platforms by their API levels. The Compile SDK set to API 27 demonstrates Andworx is capable of building application at this level. The other two parameters are just typical values, but note that to launch this application on Andworx, you will need to install the Target SDK and to exercise the support library capability, you will need to install the Min SDK along with an virtual device that is runnable on your PC.

The final paned of interest is Dependency status:

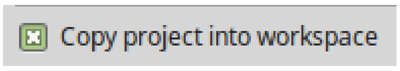


Dependency status

✓ 20 dependencies resolved

This shows 20 dependency archives have been successfully resolved. The build file only specifies 3 implementation support libraries, so the fact there is a dependency tree is clearly demonstrated. The resolution of the dependencies may take a while if files need to be downloaded. This happens in the background so the user may cancel if network issues are causing time outs.

The remaining elements of the Import Android Project dialog, are unchanged from the original ADT.



☒ Copy project into workspace

Note that you should tick "Copy project into the workspace" as recommended above.

If no errors have occurred, then the wizard "Finish" button will be enabled. Click it to commence creation of the new project in the workspace.

Project Creation

Andworx takes full control when creating the project so there is nothing to do other than watch the busy cursor. Andworx copyies source files into the workspace and creates Eclipse-specific project files such as .project and .classpath. The dependencies resolved when importing the project are automatically added to the classpath. The Android configuration is transformed to a set of Java beans which are persisted in a relational database. The project profile, which includes the project details displayed in the Project Import Wizard, are stored in the same way.

The source files are copied to the workspace location without modification as Andworx follows Gradle's source set defaults. The workspace project also has written to it a small text file named "project_ID" which identifies a the project in the database.

Project Development Build

As soon as a project is imported into Andworx, it undergoes a full debug build. This will take a while, partly due to some first-time initialisation. If you take a look at the contents of the project build directory, it should be mostly identical to one formed in the Gradle build of the same project. You should observe deprecation warnings generated by the compiler. These are normal and are a side effect of what the example is trying to demonstrate.

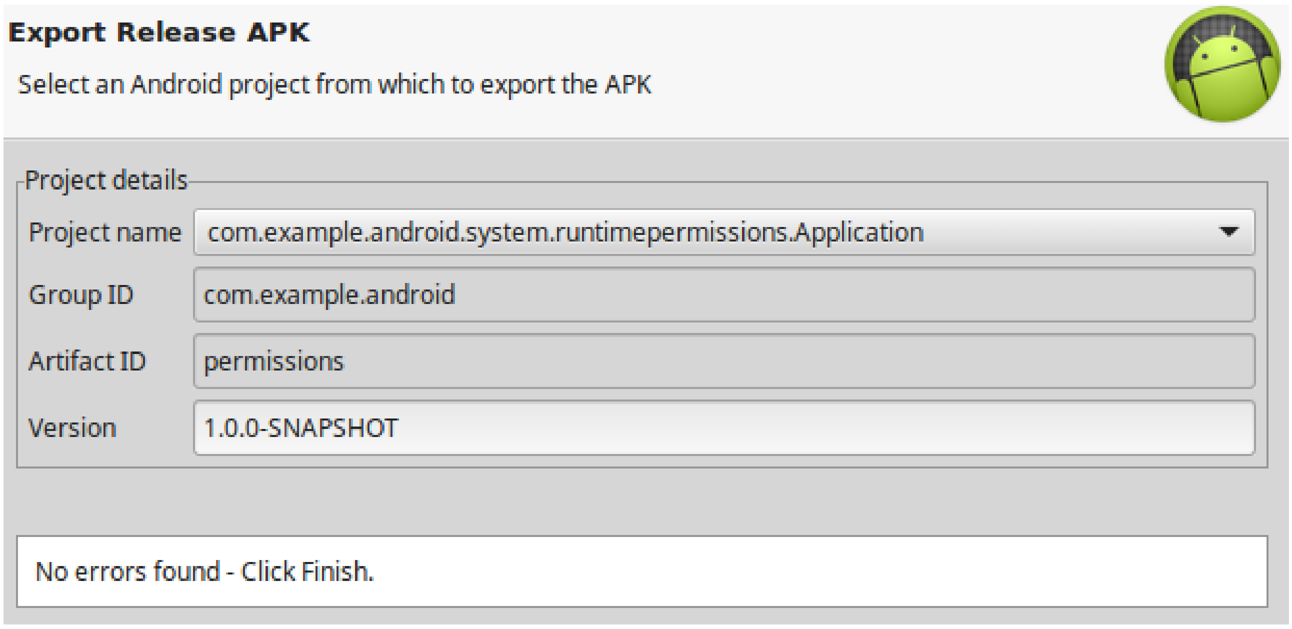
Note that linking of error markers to sources and other reporting functions are yet to be implemented. Lint check can also be launched, but currently generates spurious errors due to a bug with obtaining the minimum SDK value.

Project Export

Andworx is capable of performing release builds which includes Pro-Guard compression and digital signing of the APK. This is initiated by performing a project export operation. If you have not already done so, perfrom the following preparations:

1. Have a private key for signing the APK available, stored in a keystore file.
2. Have an attached device or emulator available for running the release APK

Right click on the Permissions project in the project explorer window and select Export ─► Android ─► Export Android Application ─► Next. This brings up the "Export Release APK" wizard. Following is the upper portion of the export wizard that you will see if the signing configuration is valid. The Finish button will be enabled, so click it to initiate the release APK build.


Export Release APK

Select an Android project from which to export the APK

Project details

Project name	com.example.android.system.runtimepermissions.Application
Group ID	com.example.android
Artifact ID	permissions
Version	1.0.0-SNAPSHOT

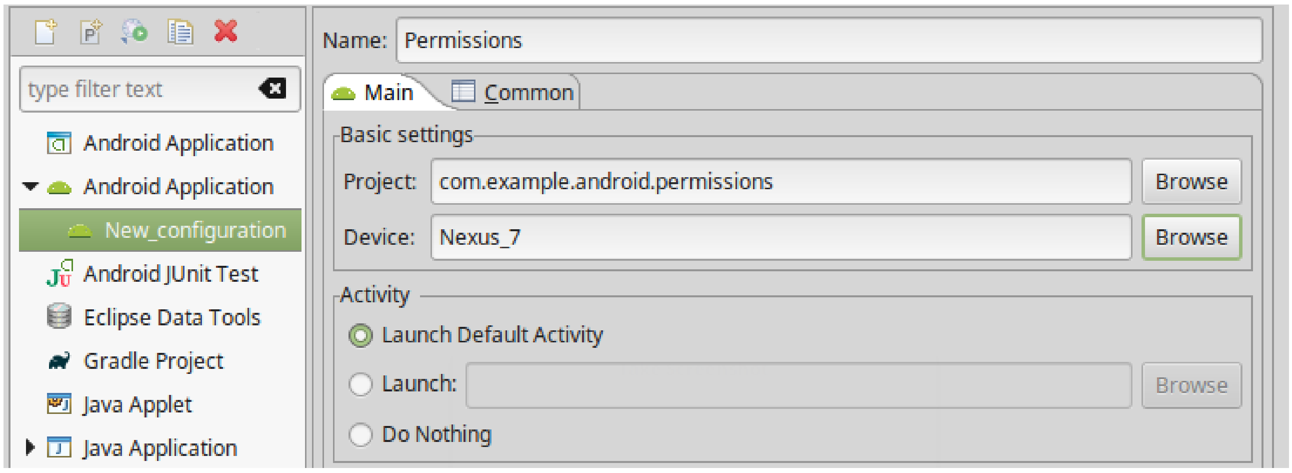
No errors found - Click Finish.

If the signig configuration is not valid, then you will see a different status message ending with " - Click Signing Button", which refers to the button just below the message. Click the Signing button to fix the configuration.

You will observe the busy cursor while the release build proceeds. When finished, a File Save dialog applears for you to choose where to save the release APK and it's name. The exported application is easily tested by installing the APK on an Android device, which will reject it unless it is correctly signed.

Launch/Debug Application

This operation pre-supposes that either a suitable virtual or physical device has been set up to run at the target SDK API level. There is a context menu item in the project explorer window to run an Android Application, but for the first attempt, it is better to configure an application launch from the "Run Configurations" dialog. For example:



Name: Permissions

Basic settings

Project: com.example.android.permissions

Device: Nexus_7

Activity

[✓ Launch Default Activity] Launch: Do Nothing

Hitting the "Run" starts the launch. If the selected device is an emulator and it needs to be started, then you will be presented with a dialog with some options such as "Wipe user data". Dismiss this dialog after hitting the "Launch" button and waiting for the progress bar to fully advance. Note that there may be a period where nothing seems to be happening, but the virtual device should eventually appear on the display. This period of no progress displayed is the result of changing the launch design and needs to be remedied.

Once the application is running, you should switch to the DDMS Perspective where both the device and application should be displayed. To debug the application, select it and then click on the toolbar icon at the top. You should now be able to set a breakpoint in the Runtime Permissions source and hit it.