Generating Signed and Unsigned Android Application Packages

IntelliJ IDEA provides facilities and interface for extracting Android application packages (.apk files). IntelliJ IDEA supports integration with the *Android Asset Packaging Tool (aapt)* which compiles the application resources. For more details, see http://developer.android.com/tools/building/index.html.

By default, *aapt* compresses resources during packaging. To have resources of a certain type included in the `.apk` file uncompressed, type `-0 <file extension for this type of resources>` in the **Additional command line parameters:** text box on the **Compiler** tab of the Android Facet dialog box. As a result, all the files with the specified extension will be excluded from compression.

With IntelliJ IDEA, you can extract both *signed* and *unsigned* `.apk` files:

- Extract *signed packages* to deploy and run them on physical devices. Based on this signature (certificate), the Android system identifies the author of a deployed application. You do not need to apply for a personal signature to any authority, a self-generated signature is quite sufficient. With IntelliJ IDEA, you can extract a *signed package* in two ways:
  - Use the Extract a Signed Android Application Package Wizard. The package will be signed during extraction.
  - Configure the `.apk` file as an artifact by creating an artifact definition of the type *Android application* with the **Release signed** package mode. When IntelliJ IDEA builds the package in accordance with this definition, the package is signed it automatically.
- Extract *unsigned packages* to test them on emulators. Unsigned packages can be extracted only through artifact definitions with the **Release unsigned** package mode specified.
- Extract and sign *debug* packages. This signature is sufficient for testing and debugging applications but does not allow publishing them. Signing packages in the debug mode is available only through an artifact with the **Debug** package mode specified.

You can also have the application obfuscated during packaging through integration with the ProGuard built-in tool.

In this part:

See Also

Concepts:

- Artifact

Procedures:

- Generating a Signed Release APK Using a Wizard
- Configuring Artifacts
- Android

Reference:

- Android Tab
- Artifacts

External Links:

- http://developer.android.com/tools/publishing/app-signing.html#signing

Web Resources:

- Developer Community