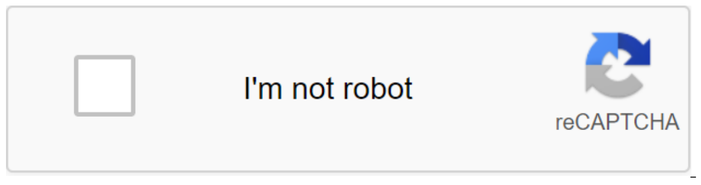

Continue

Apk file editor

APK Editor is an app that lets you do what your name says. Edit any APK that is stored on your device, and if you don't have an APK, you can pull it from any app you've installed with. Add or delete languages or even delete permissions, and it all depends on how qualified you are. But if you can't use it, it can be a lot of fun, but either way, you can find examples and instructions to use the app properly on the app's help page directly. Today I would like to share with you my findings on how the existing .apk file can be changed. File .apk is a mobile app as it is installed on a mobile device like smartphone, tablet, wearable, etc. with any packager like for example WinRAR so you can easily open it and view the files - although browsing most of the files won't make you happy, because you'll understand that they are made up, in binary format, etc... but that's another story. In any case, you can open the archive and then change any resource file and keep the modification in the archive. But if you then try to install .apk on your smartphone (or tablet or similar), you get a bug. The following screenshot shows an error in installing a modified sample of the myApp.apk app on an Android device: The reason is that after the modification, the check and signature are no longer valid. Thus, a simple change to the .apk file is not possible. However, there is still a valid case of use to modify or replace files inside the existing .apk. For example: - files that are placed in an asset folder - property files containing configuration data - images that can be replaced - stacking information resources and the like. My personal use case was: I created an Android app using the SAP Netweaver Gateway Performance accelerator. I have to apply to my users as a .apk file. But there was a requirement that they want to change the finished application (configuration change data). So I had to figure out how to achieve this: change the application without having access to the source code. Below I share with you the necessary steps. The description is based on the following software and versions: Android current API 19 Java 7 Windows 7 If you are not familiar with Android but want to be, you can check the documents and all the prerequisites for understanding this blog are explained there. Note: To run the commands below, you need to have Java on your Path variable of your Windows system (see 1 for an explanation). Review There are 3 steps that need to be followed in order to change the existing .apk file: 1. Make the actual desired changes inside the .apk 2 file. Sign .apk 3. Set .apk on device 1. Change the resource in the .apk Open .apk file with WinRAR (if that doesn't work, rename the .apk file extension to .zip) Change the resource in the archive at will (packager tools allow you to modify files without having to extract the archive) Once you are done with your changes, you should take care of the signature files that are part of .apk : Inside the archive go to the FILE- RSA and K. SF files The next screenshot shows the contents of the META-INF folder in the .apk file: Now the archive can be closed. In case you changed the file extension before, you now have to change it back to .apk 2. You don't allow you to sign .apk Android app (apk) that isn't signed. When developing an app in Eclipse ADT The Developer's Tools, an extension of Eclipse that supports development for Android) takes care of signing an app with a default certificate before installing on the device. This is convenient, but with the following description, anyone can sign an application. Signing .apk is done in 2 steps: (a) Create a certificate b) sign .apk with the created certificate Both steps are made with commands on command line (a) Create a certificate If you work in java, you have JDK in the file system. JDK comes with a certificate management tool: a keyboard. You can find it in the folder .../bin your JDK installation. Example: On my car it's here: Now you can create a certificate using the below command. However, before you do it, please check the notes below, in order to adapt the parameters of keytool.exe -genkey-v-keystore-lt;myKeystore-gt;-alias-lt;myAlias-gt;-sigalg MD5withRSA -keyalg RSA -keysize 2048 -reality 1000 Please please Please note that you have to tailor some of the options of the above team to your personal needs: keystore here, you can provide an arbitrary name for your keystore. The name you provide here will be the keystore name that will be created. The file will be created in the current catalog. (I haven't tried it, but you can probably enter the name of the existing keystore file, in order to keep the new certificate there) alias here too, you can provide an arbitrary name for a pseudonym. It is designed for you to recognize it. A pseudonym is the human readable name of the certificate that will be created and stored in the key store. 1000 This is the number of days desired. You can enter any number you wish. I think it should be high enough to avoid expiration problems. Note that sigalg and keyalg options require JDK 7, so you shouldn't need to add them if you use JDK 6 Example: keytool.exe-genkey-v-keystore mykeystore -alias myAlias -sigalg MD5withRSA -keyalg RSA -keysize 2048 -10000 when performing commands, You will get a few hints on the command line, asking for a password, username, organization, city, etc. you can enter any arbitrary data here, you just have to make sure to remember the password. Once you've completed the command in the current directory (from where you made the command) you'll see the generated keystore file in the file system (from where you made the command) you can now start signing .apk using a newly created certificate. b) Sign the apk Before signing the .apk file, you should make sure that .apk has no certificates. This is described in step 1 above. To sign the archive, we use the jarsigner tool provided by JDK which can be found in the same place as the keyboard. The next team is used to sign the team. Apk. jarsigner -verbose -sigalg MD5withRSA -digestalg SHA1 -keystore Please please note that you have to tailor some of the parameters of the above team to meet your personal needs: keystore Here you have to enter the name you gave at the previous stage a) In order to keep the command line short, I recommend temporarily copying the keystore file in the same place where you are qlt;lt't'tt; zlt;appName'gt; qlt'lt'lt.lt.gt; Here you have to enter the name of the apk file that you want to sign In order to keep the command line short, I recommend temporarily copying the .apk file in the same place where you execute the command. Here you have to enter the name of the alias that you provided when creating the certificate Note that sigalg and digestalg options are required by JDK 7, so there should be no need to add them if you use the JDK 6 Example: jarsigner -verbose -sigalg MD5withRSA -digestalg SHA1 -keystore myApp.apkAlias Once you have completed the command, you can check the result inside the file .apk: Open the archive, go to the folder .../META-INF and check if the files are CERT. RSA and CERT. SF were created. 3. Install the apk on the device Now that the .apk file is signed, you can install it on the device. BTW: This procedure is also called side load. For android applications, the installation is done on a command line using the adb command. ADB means Android Debug Bridge adb.exe is part of the software that connects the COMPUTER with the Android device. This allows you to access the device, allows you to initiate operations, transfer files, etc. In order to install .apk on the device, you must connect the device to the computer via a USB cable, and then execute the following ADB commands set in order to hold the command short line, you can temporarily copy the apk file in the same place where you perform the command. Example: adb install myApp.apk Result should be the success of the message on the command hint. If not, any of the previous steps may fail. That's it. You can find the app in your smartphone's app folder. This procedure worked for me on WIN7 and JDK 7. This was not required to restore the application, or to create new checks or similar ones. Links Please refer to the following documents for a lot of information for beginners. They also contain a lot of additional links for further reading. Start with GWPA: Preconditions: Start working with GWPA: Android Preparation: amp;lt;/appName> apk sa file editor. apk file editor pro. apk file editor for pc. apk file editor online. apk file editor pro download. apk file editor for windows. apk file editor for android. apk file editor for windows 10