EB GUIDE

Installation of EB GUIDE GTF on Android

Version 6.11.0.210526170413

1. Installation of EB GUIDE GTF on Android

The Android application package (APK) file format is used to distribute and install applications and other middleware on Android devices.

1.1. System requirements

The APK version that is currently released for EB GUIDE GTF is designed to run on a wide range of Android devices.

Table 1. Minimal requirements

Architecture	ARMv7
Platform	EB GUIDE GTF: Android 9.0 (API Level 28)

1.2. Features of the EB GUIDE GTF SDK for Android

Table 2. Features of the EB GUIDE GTF SDK for Android

Feature	**Description**
Life cycle management	EB GUIDE GTF supports Android life cycle management.
Multi-touch support	EB GUIDE GTF supports up to ten fingers for multi-touch. The number of supported fingers may be limited by the Android device.
Key handling	EB GUIDE GTF processes 16-bit UTF key mapping codes.
Interaction with the Java API	EB GUIDE GTF can be started and controlled by the Android activity. Example code and a template implementation are provided by the application. A native activity is not necessary.
Android layout handling	The exported EB GUIDE model is informed through events if the layout of the visible screen area changes. That way you can handle a virtual keyboard or changes in rotation.

1.3. Files in the EB GUIDE GTF SDK for Android

The EB GUIDE GTF SDK for Android contains `EB GUIDE Launcher.apk`. The EB GUIDE Launcher starts EB GUIDE GTF and displays the exported EB GUIDE model.

1.3.1. Released APK and custom APK

EB GUIDE GTF is delivered and installed as an APK. Use either a pre-built released APK of the released EB GUIDE version or create a custom version based on the delivered Android binaries and the APK template in the SDK. For instructions on how to create customized APK, see section 2.2, “Creating your own APK”.

The following lists help you to decide whether or not you need a custom APK.

If the following applies to your project, use the released APK:

- Your project contains EB GUIDE functionality or feature demonstrations with no further extensions.
- Your project contains project-specific extensions, for example EB GUIDE GTF extensions, to be added to the exported EB GUIDE model.
- Standard access rights are sufficient for your project. The standard access rights are `read` or `write` to the external storage of the device and network access `android.permission.INTERNET`.
- Network access is automatically granted.

If the following applies to your project, use the delivered APK template:

- You need additional access rights that are not granted by the released APK version, for example `CALL_PHONE`.
- You require a customer-specific APK, for example a customer signature for APK verification or icons.
- You use Android framework features that are not accessible in the stable API of the native development kit (NDK). The NDK contains only a small subset of features and functionality which you can use with the Java API.
- You need additional Android application functionalities that require modifications to Java-related code pieces, for example activities, services, skins, intents, or compositing.

1.3.2. Restrictions

The EB GUIDE GTF SDK for Android that is currently released for EB GUIDE GTF has the following restrictions:

- The exported EB GUIDE model is informed about rotation changes and layout changes, for example an incoming virtual keyboard on the display. The exported EB GUIDE model must handle these events.
- If the system uses Android layout handling, the Android flag `SOFT_INPUT_ADJUST_NOTHING` must not be set in the configuration of the Android activity.

1.4. Android life cycle management

The Android life cycle management is an optimization implemented by the Android operating system. If an application moves to the background, Android releases all graphics resources and makes the resources available for the application that moves to the foreground. An application is responsible for recreating the resources when it moves to the foreground.

1.5. Folder for EB GUIDE models

The exported EB GUIDE model is stored in `Android/data/com.elektrobit.guide_model_launcher/files`. This folder is automatically created when the EB GUIDE Launcher is installed and contains the exported EB GUIDE feature demo model. If you uninstall EB GUIDE Launcher, this folder is removed.

1.6. Android layout handling

Android is designed for mobile devices. On a mobile device, some characteristics concerning the layout of the visible screen area need to be considered.

EB GUIDE provides events that indicate layout changes in the visible screen area.

Example 1.
Examples for layout handling

- When a mobile device is rotated, the GUI has to adapt according to the rotation.
- When a virtual keyboard is displayed on the screen, the GUI has to adapt to the new element.

2. Using and creating an APK for EB GUIDE GTF

For background information on APK, see section 1, “Installation of EB GUIDE GTF on Android”.

For more information on Android setup, APK creation, or the Android toolchain, refer to the official Android documentation.

As the basic concepts and approaches known for other platforms are also valid for the Android platform, the following sections focus on the topics that are specific for Android.

2.1. Executing an exported EB GUIDE model on Android

Executing an exported EB GUIDE model on Android

To execute an exported EB GUIDE model on Android, you install the EB GUIDE Launcher. The EB GUIDE Launcher executes an exported EB GUIDE model on the Android device. By default, EB GUIDE Launcher contains the exported EB GUIDE feature demo model.

Prerequisite:

- To install the application on the Android device, enable your system to install from a different source than the Android Play Store. On your Android device select the **Settings** > **Security** > **Unknown sources** option.

Step 1
Copy `EB GUIDE Launcher.apk` from the `$GTF_INSTALL_PATH/platform/android/bin/` folder to your Android device or to the external storage of your Android device.

Step 2
Open a file manager and navigate to the copied file.

Step 3
Install `EB GUIDE Launcher.apk`.

Step 4
Export an EB GUIDE model.

For more information, see EB GUIDE Studio user guide.

Step 5
Copy the whole folder that was exported by EB GUIDE Studio to your Android device. For information where to store the EB GUIDE model, see section 1.5, “Folder for EB GUIDE models”.

Step 6
To execute the EB GUIDE model on your Android device, open `EB GUIDE Launcher.apk`.

The `EB GUIDE Launcher.apk` is started automatically with the EB GUIDE model which is found in the dedicated folder. The EB GUIDE model is executed on your Android device.

2.2. Creating your own APK

The APK files installed with the Android SDK of EB GUIDE GTF are suitable for most use cases. But you can integrate additional EB GUIDE GTF extensions that are useful for a project. Save the additional EB GUIDE GTF extensions in the folder of the exported EB GUIDE model and include them in the start-up configuration file. All run-time dependencies are resolved by EB GUIDE GTF.

To create a customized APK, you can use the APK template. For instructions, see section 2.2.1, “Creating your own APK using the template”.

You can also create your own APK from scratch. For instructions, see section 2.2.2, “Creating your own APK from scratch”.

2.2.1. Creating your own APK using the template

Creating your own APK using the template

Prerequisite:

- Android Studio is installed.

Step 1
Import the project `$GTF_INSTALL_PATH/platform/android/apk/GtfAndroidAppTemplate` into Android Studio.

Step 2
Optional: To change the location of the EB GUIDE model and the libraries, edit the implementation of the template `TemplateActivity.java`.

The template activity is the main activity of your custom application.

Step 3
In `$GUIDE_INSTALL_PATH/projects/code/apk/AndroidAppTemplate/src/main` create the folder structure `jniLibs/armeabi`.

Step 4
Copy the Android SDK binaries delivered with EB GUIDE GTF to the folder `$GTF_INSTALL_PATH/platform/android/src/main/jniLibs/armeabi`.

Step 5
Run `AndroidAppTemplate` in Android Studio on the target device or use an Android Virtual Device (AVD).

The `com.elektrobit.gtf_android_template` folder is created.

Step 6
Copy an EB GUIDE model to the default external file folder of the application. The default folder implemented in the template activity is `Android\data\com.elektrobit.gtf_android_template\files`.

Step 7
Deploy and launch the application in Android Studio on the target device or use an Android Virtual Device (AVD).

The EB GUIDE model is executed on your Android device. Customize the application according to your requirements.

2.2.2. Creating your own APK from scratch

Creating your own APK from scratch

Prerequisite:

- Android Studio is installed.
- Gradle Build Tool is installed and Gradle Wrapper is generated. Alternatively, you can copy the `gradle` folder and the following files from the APK template:
  - `gradlew`
  - `gradlew.bat`
  - `build.gradle`

Step 1
Create an Android project in Android Studio.

Step 2
Add the library `$GTF_INSTALL_PATH/platform/android/bin/GtfBridge.aar` to your Android Studio workspace.

Step 3
In the project workspace, create a folder `jniLibs/armeabi` and copy the `.so` files from `$GTF_INSTALL_PATH/platform/android/bin/` into the folder.

Step 4
Add the import `com.elektrobit.gtf.android.GtfActivity`.

Step 5
Create an activity that extends `GtfActivity`.

Step 6
Adapt the following methods:

Step 6.1
To set the model path, call the method `protected String getModelPath()`.

Return either `getStandardModelPath()` for the default path, or a string with the path of the EB GUIDE model files.

Step 6.2
To load additional libraries, add their names to the string array, which is the return value of `protected String[] getAdditionalNativeLibs()`. If you do not use any additional libraries, return `null`.

Step 7
In the manifest, modify or add the following code:

```
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
```

```
<uses-permission android:name="android.permission.INTERNET"/>
```

Step 8
In `build.gradle` replace the code for `applicationId "com.elektrobit.gtf_android_template"` with the name of your project.

Step 9
Create a keystore file for a release build.

Step 10
Create the release build with Gradle Build Tool on the command line or inside Android Studio.

TIP

Debug builds

Test and create debug builds within Android Studio. The Android Studio plug-in takes care of the whole APK build process, for example debug keystore.

TIP

Sub-projects for a separation of functionality

Place the activity in a master project and divide individual functionality into sub-projects. The master project references the other components as sub-project dependencies and the Android build performs the necessary integration steps.