Dictionary
App Builder

Building Apps

SIL

Dictionary App Builder: Building Apps

Last updated: 17 July 2024

Contents

1. Preparing content for your app

Before you build an app with Dictionary App Builder (DAB), you need to get your content (lexicon file, images and audio) into formats that DAB can handle.

1.1. Preparing your lexicon file

DAB can read two types of lexicon database file: LIFT and XHTML.

- **Lexical Interchange FormaT (LIFT)**
  LIFT files can be exported from any of the following SIL dictionary programs: FieldWorks Language Explorer (FLEx), WeSay or Lexique Pro.

  In FLEx, select **File** ➢ **Export**, then choose 'Full Lexicon (LIFT)'.

- **XHTML**
  DAB supports XHTML files which have been exported from FieldWorks Language Explorer (FLEx). XHTML files generated from other sources are not supported.

  In FLEx, select **File** ➢ **Export**, then choose 'Configured Dictionary (XHTML)'.

  You will also need to export a Reversal XHTML file for each index language. To do this, select **File** ➢ **Export**, then choose 'Reversal Index (XHTML)'.

For more information about SIL dictionary software, please refer to the following websites:

LIFT	https://code.google.com/p/lift-standard/
FLEx	http://fieldworks.sil.org/flex/
WeSay	http://wesay.palaso.org/
Lexique Pro	http://lexiquepro.com/

1.2. Preparing images

Images should be in JPEG or PNG format.

Keep the image size small enough so that they display well on a small screen and will not make the app size too large. DAB will allow you to resize the images after you have added them to the app project.

1.3. Preparing audio

If you want to include audio files in your app, these need to be in MP3, WAV or 3GP audio format.

Keep the audio files at a size where the quality is good enough for a phone and where the file size is not too large.

2. How to build your first app

To build your first app with Dictionary App Builder:

1. Launch **Dictionary App Builder** from its icon on the desktop.

2. Click **New App** on the toolbar. The New App wizard will appear.

3. On the first page of the wizard titled **Lexicon Database**, click **Browse…** and select the lexicon data file you want to display in the app.

   Click **Next** to move to the next page.

4. On the next page of the wizard titled **Lexicon Details**, you will see the number of entries and the languages found in the lexicon.

   Click **Next** to move to the next page.

5. If your lexical database is an XHTML file, the next page of the wizard will be titled **Reversal Indexes**. Click **Add Reversal Index File…** and select one or more reversal index files that you have exported from FLEx.

   Click **Next** to move to the next page.

6. On the page of the wizard titled **App Name**, specify the **App Name**, such as “Dogon Dictionary”, “Mamara Lexicon”, etc.

   This is the main title of your app and will be seen by the user. Do not include underscores or hard to understand abbreviations.

   Click **Next** to move to the next page.

7. On the page of the wizard titled **Package**, specify the **Package Name**, a dot-separated string which uniquely identifies your app.

   More details about choosing a good package name can be found in section *4.1. How should I choose the app package name?*

   Click **Next** to move to the next page.

8. On the page of the wizard titled **Indexes**, select the languages for which you would like to see an index tab in the app.

   Click **Next** to move to the next page.

9. On the page of the wizard titled **Font Handling**, you can select GeckoView if you know that the standard Android components will have trouble displaying the text correctly (e.g. if it is a complex script). More information on GeckoView can be found in the **Fonts** section of this document.

Click **Next** to move to the next page.

10. On the page of the wizard titled **Fonts**, choose the font for each language. You can either select from the given list of fonts or click **Other** to specify a different TrueType font file.

    Click **Next** to move to the next page.

11. On the page of the wizard titled **Color Scheme**, choose the color scheme for the app. The color you choose is the one that will be used for the main app bar. Individual colors for text, titles, links, backgrounds, etc. can be customised later.

    Click **Next** to move to the next page.

12. On the page of the wizard titled **Icon**, choose the application launcher icon. You can select one of the images in the table or if you have your own PNG image files for the icon, click **Browse** and select them.

    Click **Next** to move to the next page.

13. On the page of the wizard titled **Signing**, you need to specify the keystore and alias to use to sign the app. An app must be signed in this way so that it can installed on an Android device.

    If you do not already have a keystore file (which you are unlikely to have if this is your first time using the program):

    i. Click **Create KeyStore**.

    ii. Enter a new filename for the keystore, such as “keystore1” or something like that. Specify a password. Click **Next** to continue.

    iii. Enter an alias name for a key to create within your new keystore, such as “key”. Specify a password, which can be the same as the password you entered on the previous page. Click **Next** to continue.

    iv. On the **Certificate Issuer** page, provide details of your organisation in at least one of the fields. Click **Next** to continue.

    v. A new keystore will be created for you. Click **Close**.

14. Back on the **Signing** page of the New App wizard, you need to specify the keystore password, select the alias and enter the alias password (just as you entered them in the step above).

    Click **Next** to continue.

15. On the page of the wizard titled **Project**, you can enter modify the project name and add an optional description of the app project. Neither of these will be visible to the user of your app. They are just for your own use and might help you distinguish between multiple app projects.

    Click **Next** to continue. The New App wizard will close and the app definition will be added to the tree view on the left of the screen.

16. Take a look at each of the app configuration pages by selecting them in the tree view on the left. Look in each of the tabs on each page to verify that you have the settings you want. You can always go back to them later to change them if you find you need to make modifications to fonts, colors, styles, etc.

17. When you have finished configuring the app, click the **Build Android App** button on the toolbar at the top of the screen.

    If something is not configured correctly for the build to work, you will be notified of this.

18. A black command box will appear. Wait about a minute while the app is compiled.

    The first time the build process is run, the compiler needs to connect to the internet to download some files. After this, subsequent app builds will not require internet access. See **Tools ➢ Settings… ➢ Build Settings** to turn on offline mode after the first app build.

19. If the build succeeds, you will have a new APK file – the installation file for an Android app.

The next section describes how to copy this APK file to your phone and launch the app.

3. Installing the app on your phone

In the above section, you have seen how to compile an Android app. The result is an APK file, the installation file for an Android app. You now need to copy this APK file to your phone, install it and launch the app.

Here is how to do this:

1. Connect your Android phone to your computer using a **USB data cable**.

   (Sometimes you get cheap USB cables that can only charge a phone but cannot transfer data, so make sure you have the right kind of cable.)

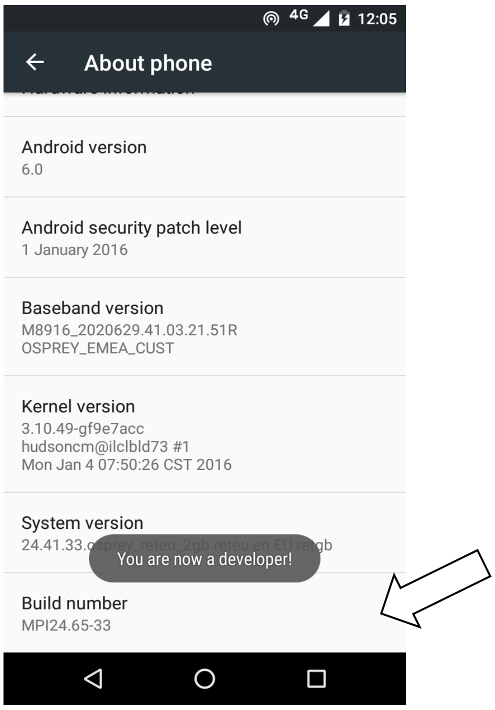
2. Ensure that **Developer Options ➢ USB Debugging** is enabled on your phone. By default, on new phones, Developer Options is turned off. This is how you can enable it:

   i. Open the **Settings** menu of your phone.

   ii. Scroll down to the bottom of the menu and tap on **About Phone**.

   iii. Find the **Build Number**. This could be on the About Phone page, or under a sub-menu such as 'Software Information'.

   iv. Tap on the Build Number **seven times**. As you do this, you will see a series of messages appearing: "You are now 3 steps away from being a developer", "You are now 2 steps away from being a developer", "You are now 1 step away from being a developer", "You are now a developer!".



Android version
6.0

Android security patch level
1 January 2016

Baseband version
M8916_2020629.41.03.21.51R
OSPREY_EMEA_CUST

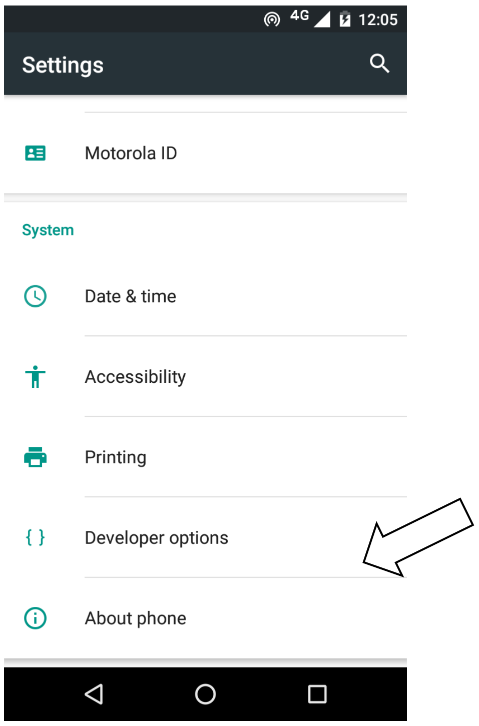
Kernel version
3.10.49-gf9e7acc
hudsoncm@ilclbld73 #1
Mon Jan 4 07:50:26 CST 2016

System version
24.41.33.osprey_reteu_2gb.reteu.en.EU retgb

You are now a developer!

Build number
MPI24.65-33

   v. Now return to the Configuration menu of your phone. Look for the Developer Options menu item. You might see **Developer Options** above the **About Phone** menu item. If you do not see it here, it could be in **System** settings, under **Advanced**. Different phones place Developer Options in different places, so look around your Configuration menu until you find it.


12:05

Settings

Motorola ID

System

Date & time

Accessibility

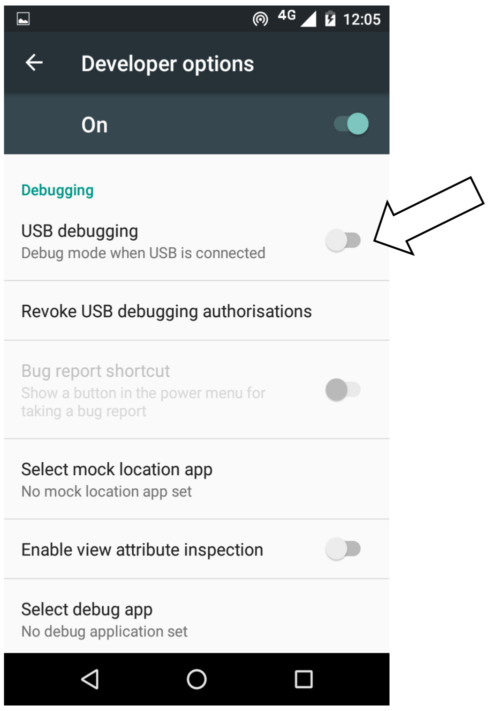
Printing

Developer options

About phone

vi. Tap on **Developer Options** and ensure that it is turned on.

vii. Scroll down the Developer Options page and find **USB Debugging**. Enable this setting.


12:05

Developer options

On

Debugging

USB debugging
Debug mode when USB is connected

Revoke USB debugging authorisations

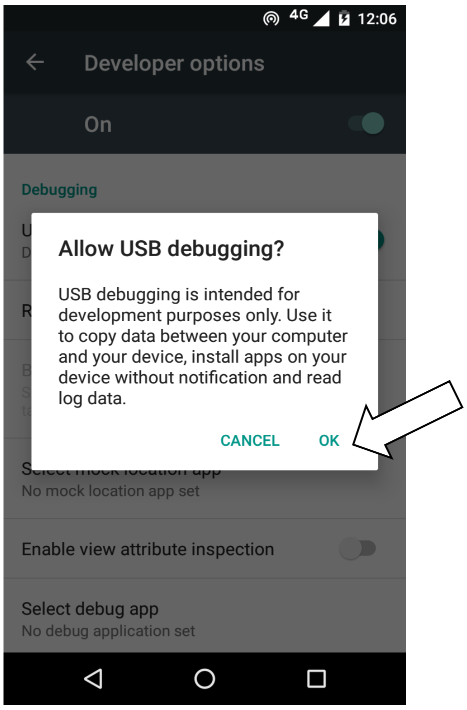
Bug report shortcut
Show a button in the power menu for taking a bug report

Select mock location app
No mock location app set

Enable view attribute inspection

Select debug app
No debug application set

When you do this, you will probably get a message “Allow USB Debugging?”. Tap **OK**.


Developer options

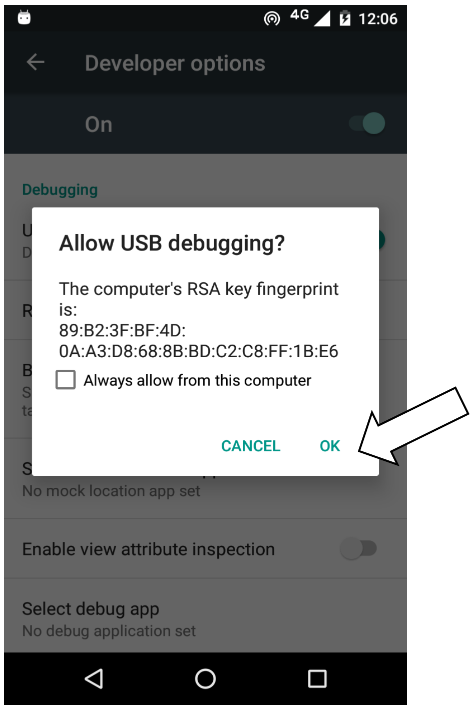
On

Debugging

Allow USB debugging?

USB debugging is intended for development purposes only. Use it to copy data between your computer and your device, install apps on your device without notification and read log data.

If you see a message box like this, tap **OK**:


Developer options

On

Debugging

Allow USB debugging?

The computer's RSA key fingerprint is:
89:B2:3F:BF:4D:
0A:A3:D8:68:8B:BD:C2:C8:FF:1B:E6

☐ Always allow from this computer

3. In Dictionary App Builder, click the **Install APK** button on the toolbar at the top right of the screen.

   A command window will open and the APK file will be copied to your phone, installed and the app will be launched.

If this does not work, look at the command window to see if there is an error message. If you see a message such as “No devices/emulators found”, it means that your phone and computer are not connected correctly or that you have not enabled USB debugging on your phone.

Note:
Described above is a two-step process: **Build App** and then **Install APK**. If you prefer, you can tell Dictionary App Builder to do this in one step, i.e. for the APK to be installed and launched automatically after building an app. See **Tools** ➢ **Settings…** ➢ **After Build** to enable this feature.

4. App Creation Basics

4.1. How should I choose the app package name?

The standard for an app package name is to begin with the reversed web address of the publishing organisation, e.g. if it is SIL, the package name could begin with:

org.sil

and will be followed by something identifying the language and type of publication, e.g.

org.sil.cccc.nnnn.lexicon

where ‘cccc’ is the country name and ‘nnnn’ is the language name.

If you work for a university or linguistics organisation, you might have standards to follow for package names, so please contact your digital publications coordinator for advice on this.

Once you publish your app on an app store, you cannot change its package name later if you want users to continue to receive updates. The package name uniquely identifies the app in the Android world. Those who install the app will be able to find its package name on their device. It will also appear in the web address for your app if you make it available on Google Play.

If you are building apps for **test purposes** on your devices, you can use a package name beginning with com.example, e.g.

com.example.test.app123

But remember to change it before you publish the app.

4.2. Do I have to create a new keystore for each app, or can I reuse the same keystore for several of my apps?

You can use the same keystore and key alias for all or several of your apps.

See here for more details:

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

4.3. Can I build apps when I do not have internet access?

The first time you build an app, you will need to be connected to the internet otherwise the compiler will fail. After that you can set the 'offline' version in **Settings** so you can work offline.

4.4. Can I build an app from the command line?

Yes, Dictionary App Builder has a command line interface which allows you to create a new app and build it, or load an existing app and build it.

The command line tool is named **dab** and can be found in the Program Files folder, usually c:\Program Files (x86)\SIL\Dictionary App Builder.

dab takes the following parameters:

Option	Description
`-new`	Create a new app project
`-load <project>`	Load an existing app project
`-build`	Build app project (use with either -new or -load)
`-no-save`	Do not save changes to app (use with -load)
`-?`	Show command line help
`-n <app-name>`	Set app name. Enclose the name in "double quotes" if it contains spaces.
`-p <package-name>`	Set package name, e.g. com.myorg.language.appname
`-i <filename>`	Include additional parameters file. Use the full path of the file and enclose it in "double quotes" if there is a space in the path.
`-a <filename>`	Set about box text, contained in text file. Use the full path of the file and enclose it in "double quotes" if there is a space in the path.

`-f <fontname>`	Set font name or filename, e.g. "Charis SIL Compact", "c:\fonts\myfont.ttf" The font name must be one of the items in the list of fonts in the New App wizard. For other fonts, specify the full path to the font filename.
`-ic <filename>`	Add launcher icon (one or more .png files). Use the full path of the files and enclose them in "double quotes" if there is a space in the path.
`-l <lang-code>`	Set language for menu items and settings, e.g. en, fr, es
`-ft <feature=value>`	Set a feature.
`-vc <integer>`	Set version code, e.g. 1, 2, 3, or +1 to increment the current version code by 1.
`-vn <string>`	Set version name, e.g. 1.0, 2.1.4, or use +1, +0.1, +0.0.1 to increment the current value.
`-ks <filename>`	Set keystore filename. Use the full path of the file and enclose it in "double quotes" if there is a space in the path.
`-ksp <password>`	Set keystore password
`-ka <alias>`	Set key alias
`-kap <password>`	Set key alias password
`-fp <folder=path>`	Set a folder path, e.g. "app.builder=c:\Dictionary App Builder".

Examples:

```
dab -load \"My App\" -build
```

5. Fonts

If you are using a non-Roman script or a Roman script with combining diacritics, it is possible that Android devices will not display your fonts correctly. To overcome these problems, try using the GeckoView library.

GeckoView is a viewer component from Mozilla that replaces the standard Android viewer. It can render Graphite fonts correctly.

You can configure this on the **Appearance** ➢ **Fonts** ➢ **Font Handling** page.

The required GeckoView library files will add at least 55 MB to your app size, so do not enable GeckoView unless you know you need it to display your fonts correctly.

You will find that when you build an Android APK with GeckoView, the APK size will be at least 200 MB larger than without GeckoView. This is because it contains the GeckoView libraries for four different device architectures (32-bit ARM, 64-bit ARM, 32-bit Intel and 64-bit Intel). In practice, your app users do not need to install such a large file.

- For **online app distribution**, you need to upload an AAB file (app bundle) to Google Play rather than an APK. The AAB file contains all the GeckoView libraries for all four device architectures, but when a user installs an app from Google Play, Google will create a tailored APK for them, with just the libraries and components their phone needs. The AAB file might be over 300 MB, but the actual size of the app for any user will be much smaller.
- For **offline app distribution**, you can ask for Dictionary App Builder to create multiple APKs, one for each device architecture. Do this on the **App** ➢ **APK** tab. Each of these APKs will be significantly smaller (around 55 MB extra for GeckoView). You just need to choose the right one(s) to distribute, according to the types of phones people have. Many of the phones today will use the 64-bit ARM APK. Otherwise, the 32-bit ARM APK is used for most older phones. Intel phones are less common, but it will depend on the phones being sold in your country.

6. Audio

6.1. How do I distribute the audio MP3 files with the app?

There are 3 ways of including audio files in your app: assets, external folder or internet download. You can use a single **audio source** for all of the files in an app or you can combine two or more audio sources in an app.

To specify the audio source(s) in Dictionary App Builder, you need to visit the following two tabs on the **Audio** page. This page can be found in the apps tree view just under Analytics on the top level of app pages.

1. The **Audio Files** tab, which lists the audio files with their corresponding audio source.

   To change the audio source for a file or files, select the rows you want to change and select **Change Audio Source**.

2. The **Audio Source** tab, which defines the available audio sources.

   You can modify, add and remove audio sources here.

The follow sections describe the different audio source types.

1. Assets
The mp3 files will be packaged inside the apk file for the app. This is the easiest method for a few files (e.g. one book) and requires no permissions. But be beware that the apk will get very large if you have several books of audio. The maximum size of an apk that can be uploaded to the Google Play store is 100 MB.

2. External Folder
No audio files are packaged within the app, so the apk is small. The app will look in a specified SD card folder to find the audio mp3 files it needs. If you are distributing the app via SD card, you include the folder of audio files on the SD card together with the apk. This method requires the 'Read external storage' permission but not internet access.

You can place the mp3 files inside sub-folders and sub-sub-folders in the specified SD card folder, using any folder names you choose. Alternatively, you can place all the audio files in a single folder without using any sub folders.

If the app does not find audio files in the specified folder or its sub-folders, it will also search the other folders on the device to see if it can find them there. For example, if the specified folder name is 'Audio 123' but the files are located in the 'Audio 456' folder instead, the app should find them. Once it has found a folder with a needed audio file, it will keep a note of it so it knows where to look next time.

3. Internet Download
Like method 2, no audio files are packaged within the app, so the apk is small. The app will look in a specified SD card folder to find the mp3 files it needs. If it doesn't find them there, it will look in all the other folders on the device. If it still cannot find them, the app can download the files one by one when it needs them from a website of your choice. This method requires the 'Read external storage', 'Write external storage', 'Connection state' and 'Internet' permissions.

Audio filenames
The internet download works best if your audio filenames do not include any spaces. A filename of the form “african-elephant.mp3” is better than “african elephant.mp3”.

Http or https
The download manager in Android 2.3 (Gingerbread) cannot handle downloads from secure https addresses, so if you want to support these phones, use an http:// address instead of https://.

Audio file hosting
Recommended storage locations for the files on the internet include:

1. **A language-specific website**

   If you have a language-specific website for making resources available for download, you could place the audio files in a folder on the website.

   For example, if your website is called ‘www.ourlanguage.org’, you could upload the audio files to a folder called ‘audio’. The http address for your audio files would then be: http://www.ourlanguage.org/audio

   Your website administrator should be able to help you do this.

2. **Cloud Storage Services**

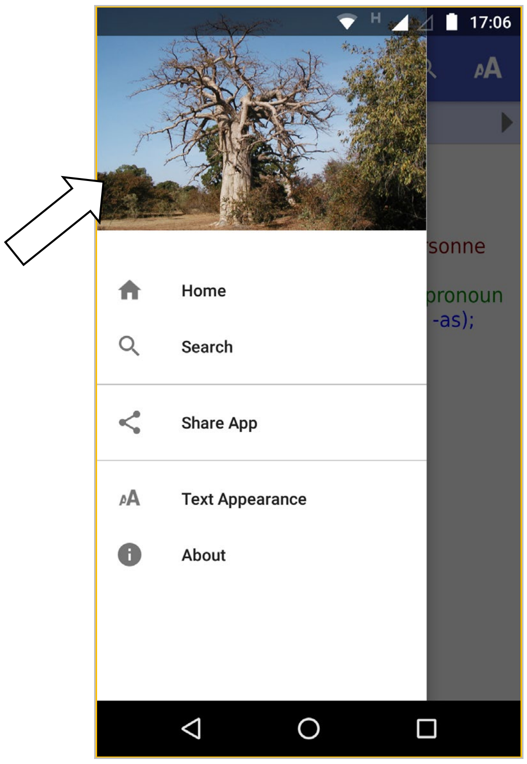
   Amazon S3 (Simple Storage Service): http://aws.amazon.com/s3/
   Backblaze B2 Cloud Storage: https://www.backblaze.com/b2/cloud-storage.html
   Google Cloud Storage: https://cloud.google.com/storage/

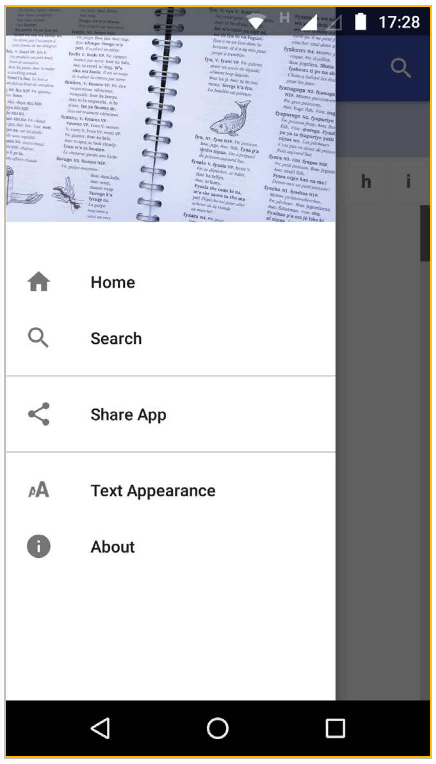
   These cloud storage services are designed for fast, reliable and secure online storage. Once you have created an account, you create a 'bucket' in which to place your mp3 files. When you add the files, you need to make them public and make a note of the web address link to use to access them, e.g. http://s3.amazonaws.com/yourbucketname

   You will get some months of free storage before there is a charge according to the bandwidth used, i.e. how many MB of audio users download. It might be easiest to organise this kind of cloud storage at an organisational level rather than creating a new account for each language.

7. Navigation Drawer

You can customise the image that appears at the top of the navigation drawer. It can be a photo, your organisation’s logo or any relevant graphic design.





Specify a landscape image file on the **Appearance** ➢ **Graphics** ➢ **Navigation Drawer** page.

8. Analytics

If you enable Analytics, the app will connect to the internet from time to time to send app usage information to one or more analytics accounts. This will give you an idea of the extent to which people are interacting with the app.

The information sent will include the model of the device (such as ‘Google Nexus 7’, ‘Samsung Galaxy S4’), the Android version (such as ‘4.2’), the mobile network provider and an approximate location (city/country). No personal information is included.

You can configure your app to send usage data to one or more of the following analytics engines:

Firebase Analytics	Sends data to a Google Firebase Analytics

	account of your choice.
Amplitude Analytics	Sends data to an Amplitude account of your choice.
S3 Digest Analytics	Sends a digest of analytics data to an Amazon S3 Bucket of your choice.

To set up analytics:

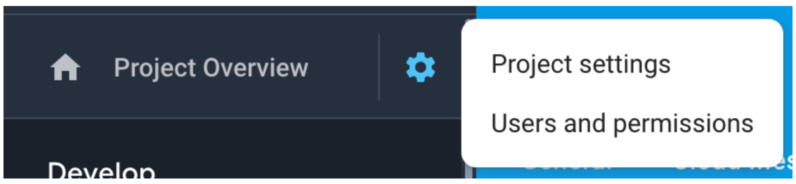
1. Go to the **Data & Analytics** ➢ **Analytics** page for the app.
2. Select **Enable Analytics**.
3. Click **Add Analytics Account…**
4. Choose an account type and enter your analytics account information.
    - For Firebase Analytics, you will need a google-services.json configuration file for your account.
    - For Amplitude Analytics, you will need an API Key
    - For S3 Digest Analytics, you will need an S3 Bucket ID and an Identity Pool ID.

8.1. Firebase Analytics

To sign up for Firebase Analytics, ensure you have a Google account.

You will need to:

1. Go to the Google Firebase website at https://firebase.google.com/ and ensure you are signed in with your Google account.
2. Click **Add project** to create a Firebase project for this app.
3. In Step 1, you will need to give your new project a **Project name**.
4. In Step 2, titled **Google Analytics for your Firebase project**, select **Set up Google Analytics for my project** and press **Continue**.
5. In Step 3, titled **Configure Google Analytics**, click on the drop-down box and choose **Create a new account**. Give it a name, which can be the same as your Firebase project name.
6. Check the boxes to confirm that you accept the analytics and data protection terms. Click **Create project** and wait a few seconds for the project to be created.
7. Click the Settings button (a cog wheel icon near the top) and select **Project settings**.



8. On the **General** tab of the Settings, scroll down to the **My apps** section and click the Android app icon.
9. On the **Add Firebase to your Android app** page, enter your app package name and click **Register app**.
10. Download the config file, **google-services.json**. That is all you need from the app registration. You can ignore the information on the rest of the screens and return to Settings.
11. Go to the **App** ➢ **Firebase** page in Dictionary App Builder.
12. Select **Firebase Analytics** as one of the features to use in the app.
13. At the bottom of the page, click the **Browse** button and find the **google-services.json** config file that you have just downloaded from the Firebase console.

8.2. Amplitude Analytics

To use Amplitude analytics, you will need to create an account. Go to:

https://amplitude.com

You will need to:

1. Click **Sign Up** at the top right of the screen and create an account. You will receive an email to finish activating your account. Copy the link to your browser and go to the page and complete the activation process.
2. You will be prompted to create a new organization. You can do that to invite team members to join your organization and access the data. You will also be prompted for some additional information.
3. Click **Create Project**, enter the project name and click **Create**. There will be a project for each individual app.
4. Click **Projects** on the left of the screen. This will show the list of projects and the properties including a long string of hexadecimal characters under the label **API Key**.
5. Highlight and copy this string into the API Key field in Dictionary App Builder.

8.3. S3 Digest Analytics

To use S3 Digest Analytics, ensure you have admin permissions to an Amazon AWS account, and go to:

https://aws.amazon.com/console/

You will need to:

1. Click **Sign In to the Console** at the top right of the screen.

2. Create an S3 Bucket.
    a. Go to the **S3** Service and click **Create bucket**, enter a Bucket name, select a Region near where the app will be distributed, and click **Next**.
    b. On the **Set properties** and **Set permissions** steps, use the defaults and click **Next**.
    c. Review the configuration and click **Create bucket**.
    d. Copy the **Bucket name** into the **S3 Bucket ID** field in Dictionary App Builder.
3. Create a Federated Identity.
    a. Go to the **Cognito** Service and click **Manage Federated Identitites**. The first time you use this service it will start creating an identity pool for you. If the AWS account already has identity pools, then it will show a grid of existing one. If this is the case then click **Create new identity pool**.
    b. Enter an **Identity pool name**, click **Enable access to unauthenticated identities** (which allows users of the app to submit analytics without logging into some service), and click **Create**.
    c. Click **Show Details** to see the **Role Name** for the unauthenticated identities (in the next step, we will give them permission to put objects in the bucket created in the previous step). Click **Allow**.
    d. Copy the **Identity pool ID** value inside the quotes in the **Get AWS Credentials** section of the **Sample code** page shown after completion of the previous step. Copy this string into the **Identity Pool ID** field in Dictionary App Builder.
4. Give permission to put data into the S3 Bucket.
    a. Go to the **IAM** Service and click **Roles** category.
    b. Click on the Role created in step #3 (e.g. Cognito_<IdentityPoolName>Unauth_Role).
    c. Click **Add inline policy**, click **Service** and choose **S3**.
    d. Click **Actions**, type in *PutObject* to search for actions, and check **PutObject** to select that action.
    e. Click **Resources**, use default Specific, click on **Add ARN** link, enter the **Bucket name** from step #1 into the **Bucket name** field, click the **Any** checkbox at the end of the **Object name** field, and click **Add**.
    f. Click **Review policy**, enter a name in the **Name** field, and click **Create policy**.

8.4. Details on S3 Digest Analytics

S3 Digest Analytics will send a small daily digest (about 300 bytes for each day the app is used) of compressed, completely anonymous data (no personal, phone, or GPS location information) via an encrypted transport (https) to an Amazon data center, when the device is connected to the Internet (via cell or WIFI) while the app is running. You are responsible for hosting the Amazon S3 Bucket and processing the received data.

Files are uploaded to the S3 Bucket and in a sub-folder based on the format (e.g. the current format is f1) and stored with a unique filename using a randomly generated GUID as the basename. We are working to package a Splunk configuration so that you can deploy your own server to analyze the data.

S3 Digest Analytics uses a JSON payload format. A complete sample data payload is below:

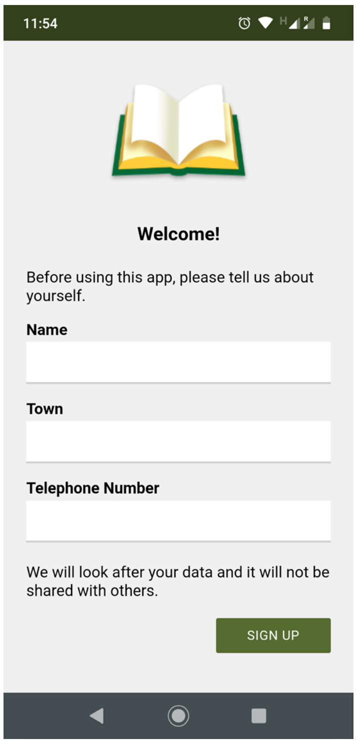
```
{"startTime":"20180306T0835Z","period":1440,"id":"12db7e3f-93d9-4370-b12b-
fe048804e4f5","package":"org.sil.dictionary.cuk","version_name":"1.0.1","s
essions":1,"sessionMins":21,"shares":3}
```

This sample is comprised of the following fields:

- One day of activity (1440 minutes), starting on 2018-03-06.
- The id is a GUID which was randomly generated on the phone when the app was first launched, enabling determination of how many unique installations of the app are in use (but no user-identifying information).
- Package and version indicate which app is in use.
- This report was for a single 21-minute session. This (and other) values would be incremented if the app had been used multiple times within the reporting period.
- The user pressed share in the app 3 times.

9. Registration Screen

You can define a registration screen to be shown when a user launches the app for the first time. This allows you to collect contact information, for example to connect people with a WhatsApp group to discuss dictionary entries.



11:54

Welcome!

Before using this app, please tell us about yourself.

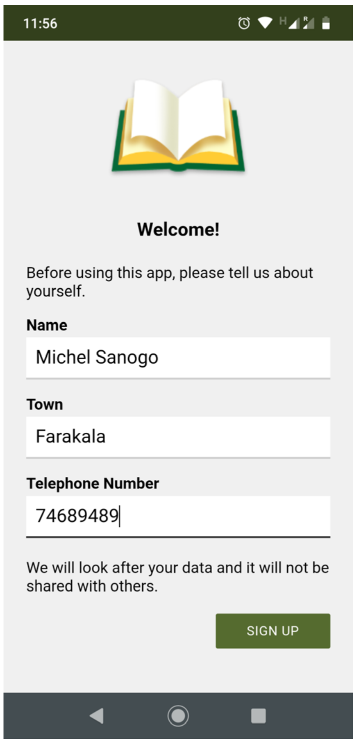
Name

Town

Telephone Number

We will look after your data and it will not be shared with others.

SIGN UP



11:56

Welcome!

Before using this app, please tell us about yourself.

Name

Michel Sanogo

Town

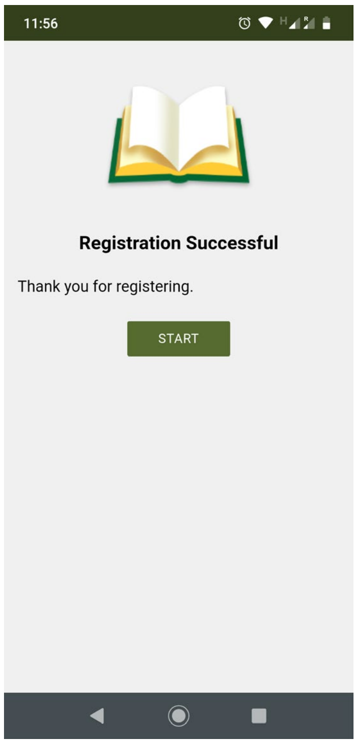
Farakala

Telephone Number

74689489

We will look after your data and it will not be shared with others.

SIGN UP



11:56

Registration Successful

Thank you for registering.

START

To set up a registration screen, you need to do some configuration work in two places:

- within Dictionary App Builder, and
- in the Google Firebase console.

9.1. Setting up the Registration Screen in Dictionary App Builder

To set up the registration screen within the app builder:

1. Go to the **App** ➢ **Security** page.
2. Select **Require each user to register with their details when they first use the app**.
3. Click the **Configure Registration** button.
4. Follow the instructions in each of the tabs in the Configure Registration dialog:
    i. **Registration Screen:** specify the title and text to show on the registration screen.
    ii. **User Details:** specify the input fields (such as name, telephone number, email adress), that you will ask the user to provide. You can specify which of these are required and which are optional.

iii. **Skip Registration:** choose whether you want the user to be able to skip the registration process. If they do press the Skip button, specify when you want the app to ask them again.

iv. **Registration Completed:** specify the title and text to be displayed on the screen which is shown to the user after they have successfully registered.

v. **Image:** specify an image to be displayed at the top of the screen, such as your organisation's logo or the app icon.

vi. **Database:** choose whether the user data will be stored in an app-specific path in the database (which is useful if you use the same database for several apps), or at the top level of the database (which is fine if you have just one app). More information about configuring your database can be found in the following section.

vii. **Styles:** you can adjust the styling of the screens by adjusting the style declarations.

Use the **Test** buttons on the **Registration Screen** and **Registration Completed** tabs to preview the screens in a browser.

5. Click **OK** when you are finished in the dialog.

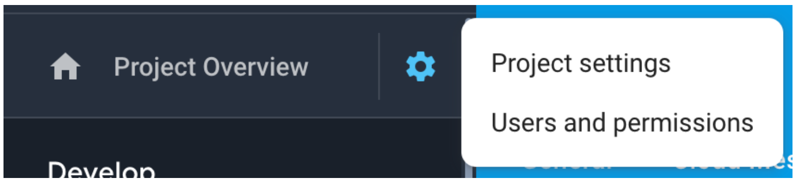
9.2. Setting up the database in the Google Firebase console

To set up the database, which will contain the registered users' information, you need to add Firebase to your app, create a database, set up authentication and configure rules.

Add Firebase to your app

To add Firebase to your app:

1. Go to the Google Firebase website at https://firebase.google.com/ and ensure you are signed in with your Google account.
2. Create a Firebase project if you do not already have one for this app.
3. Click the Settings button (a cog wheel icon near the top) and select **Project settings**.



4. On the **General** tab of the Settings, scroll down to the **My apps** section and click the Android app icon.

5. On the **Add Firebase to your Android app** page, enter your app package name and click **Register app**.
6. Download the config file, **google-services.json**. That is all you need from the app registration. You can ignore the information on the rest of the screens and return to Settings.
7. Go to the **Data & Analytics** ➢ **Firebase** page in Dictionary App Builder.
8. Select **Firebase Realtime Database** as one of the features to use in the app.
9. On the **Firebase Configuration** ➢ **Android** tab, click the **Browse** button and find the **google-services.json** config file that you have just downloaded from the Firebase console.

Create a Database

To create a **Realtime database**:

1. In your Firebase project console, select **Database** from the menu on the left of the screen.
2. Scroll down the screen for the section titled **Or choose Realtime Database** and click the **Create database** button.
3. Choose **Start in locked mode** as the Security rules and click **Enable**.

Set up Authentication

On the **Authentication** page of the Firebase console, enable two authentication types on the **Sign-in method** tab:

1. **Email/Password** (this is used when viewing registered users in a web browser)
2. **Anonymous** (this is used during user registration in the app)

Configure Rules

Rules are required to tell Firebase who will have access rights to read and write to the database. To configure the database rules:

1. In your Firebase project console, select **Realtime Database** from the menu on the left of the screen.
2. Select the **Rules** tab.
3. The rules you specify will depend on the path you have chosen on the **Database** tab in the Registration configuration.

   If you have chosen to store the user information in an **app-specific location**, e.g. “/apps/your-app-package-name/users/”, the rules need to be as follows:

```
{
  "rules": {
    "apps": {
      "$package": {
        "users": {
          "$uid": {
            ".read": false,
            ".write": "$uid === auth.uid"
          }
        }
      }
    }
  }
}
```

If you have chosen to store the user information at the **top level of the database**, e.g. “/users/”, the rules need to be as follows:

```
{
  "rules": {
    "users": {
      "$uid": {
        ".read": false,
        ".write": "$uid === auth.uid"
      }
    }
  }
}
```

These rules give write access only to the user who is making the registration. No one will have read access (except for you when you view the database from the Firebase console).

Tip: Make sure you copy the rules exactly. An easy way to get the rules you need is to click the **Database Rules** button on the Database tab. This will give you the rules you can copy.

4. Click **Publish** to confirm your changes.

Rules FAQ

Q. If we want an administrator to have read access to the Registered users to display in a web browser, what rule do we use?

If you want to give an administrator read access to the data, to be displayed in a web browser (See **Configure Registration ➢ User Details ➢ View Data**), here is the rule to use:

```
{
  "rules": {
    "apps": {
      "$package": {
        "users": {
          "$uid": {
            ".read": "(auth != null) && (auth.token.email
== 'me@abcd.com')",
            ".write": "$uid === auth.uid"
          }
        }
      }
    }
  }
}
```

(where me@abcd.com is a user added on the **Authentication** page)

The above rules assume that you want to store the user information in an app-specific location. If you have chosen the top-level location, you need to remove the “apps” and “$package” elements above.

10. Distribution

Android apps built with Dictionary App Builder can be published on the Google Play store, distributed on memory cards, shared by Bluetooth or Wi-Fi transfer, uploaded to websites, or sent out by email.

For more information, please see the user manual: *Distributing Apps*.