



Scriptoria Publishing Properties

Last Updated: 10 December 2021

Introduction	1
Android Keystores	3
Android Version Name	4
Android Version Code	4
Android App Bundles	4
Share App Link	5
Publishing HTML and PWA	6
Embedding Audio	8
Download Audio During Build	8
Change Audio Source Before Build	9

Introduction

Scriptoria builds apps and publishes content to various stores/locations (e.g. Google Play Store, S3 Bucket, or a cloud provider). Different projects will have different requirements that may need parameters to adjust how the build or publish process is performed. Starting in version 6.1 of the App Builders, these parameters are set in the project using the Publishing Properties tab in the Publishing page.

App Publishing

Scriptoria

Publish Properties

Google Play Store Listing

Publish Properties

Some apps published with Scriptoria need to have some additional properties defined. Your Scriptoria contact person will tell you if you need to enter anything here. See the [Scriptoria Publishing Properties documentation](#) for a description of the available properties.

Add Property...

Name	Value
------	-------

Click on Add Property... to add a publishing property to the project. The dropdown in the Add Property dialog has a list of known publishing properties. A different name can be typed in if necessary.

Add Property

Name

Value

BUILD_KEYSTORE
BUILD_MANAGE_VERSION_NAME
BUILD_MANAGE_VERSION_CODE
BUILD_SHARE_APP_LINK

OK

Cancel

The assigned value will be a string or a number. The number 1 will be used for true and the number 0 for false.

Android Keystores

Publishing Property: BUILD_KEYSTORE

Android apps must be signed with a keystore. Each keystore has associated with it the keystore password, the key alias, and the key password. Scriptoria stores these together in a secure place as 4 separate files:

- *keystore_name.keystore*
- ksp.txt - keystore password
- ka.txt - key alias
- kap.txt - key (alias) password

If an Android app is published to Google Play, then updates to the app must be signed with the same keystore that was originally used to publish the app. The preferred way to publish apps to Google Play through Scriptoria is to share a keystore for all apps in an organization. However, some organizations may want to add apps to Scriptoria that were already published with a keystore other than the primary organization keystore. One common scenario is that an app is transferred from one organization to another.

When the Scriptoria Administrator adds an organization to Scriptoria, a primary keystore will be provided by the Organization Administrator which will be used by default for app builds. Additional keystores can be added to an organization by the Scriptoria Administrator. When an additional keystore should be used, then set the BUILD_KEYSTORE publishing property to the name of the *keystore_name*.

Example 1: Multiple Keystores

An organization has been publishing apps to their Google Play Store account. There were three separate people publishing apps and each person had their own keystore. Now the organization wants to use Scriptoria to publish all these apps. The Organization Administrator will provide the primary keystore for all new apps published. They will also provide user1.keystore, user2.keystore, and user3.keystore (along with the associated ksp.txt, ka.txt, and kap.txt for each). The Scriptoria Administrator will save these in separate folders based on the keystore name (e.g. user1, user2, and user3). Then these keystore names will be used in the individual projects depending on which user was maintaining the project. So if project42 was being managed by user3, then the BUILD_KEYSTORE publishing property would be set to user3.

Example 2: Transfer App

An app was migrated from the Wycliffe USA store to the Kalaam IPS Cameroon store. When the app is updated in the future through the Kalaam account, the app needs to be signed by the original Wycliffe USA keystore. However, it is important to protect the original keystore so that it

isn't accidentally copied by someone else. The Scriptoria Administrator was informed about the transfer and copied the keystore and the credential files to a sub folder of the kalaam keystore files. Then, the Scriptoria Administrator informed the user to set the BUILD_KEYSTORE publishing property to wycliffeusa.

Android Version Name

Publishing Property: BUILD_MANAGE_VERSION_NAME

The Android version name is the user visible version information shown in Google Play and in the Android Settings for the app. When Scriptoria was used only by SIL, we decided that it was best to manage the version name so that it matched the version of Scripture App Builder used to build the app. This helped the maintainer of multiple apps to know which apps needed to be rebuilt when there was a new version of Scripture App Builder. If you would rather have Scriptoria always use the version name that is specified in the App Builder project file, then set the value of the BUILD_MANAGE_VERSION_NAME publish property to the number 0.

Android Version Code

Publishing Property: BUILD_MANAGE_VERSION_CODE

The Android version code is used by the Android system software to control upgrades. Only apps with a larger version code can be used to upgrade an existing app. When Scriptoria was used only by SIL, we decided that it was best to manage the version code so that the number was always increasing. Scriptoria remembers the version code for each build that it creates. This helps the maintainer so that they don't have to remember before syncing an app to Scriptoria to make sure to change the version code in the App Builder. [Note: It also makes sure that the version code is always larger than the one specified in the project file.] If you would rather have Scriptoria always use the version code that is specified in the App Builder project file, then set the value of the BUILD_MANAGE_VERSION_CODE publish property to the number 0.

Android App Bundles

Publishing Property: BUILD_ANDROID_AAB

Starting in August 2021, [Google Play will require](#) that all new apps are published as Android App Bundles (AAB) instead of APK. To have Scriptoria build an AAB, set the value of BUILD_ANDROID_AAB to the number 1. An APK will still be built to enable testing the build on a test device and off-line distribution.

Publishing Property: BUILD_EXPORT_ENCRYPTED_KEY

If the app has been uploaded as an APK and released and later you would like to opt-in to AAB, then you will need to upload the key used to sign the app so that Google Play can sign when delivering the app to the end user's device (see [Use Play App Signing](#) for more information). Scriptoria can export the key used to sign the app in a format that only Google Play can decrypt and is safe to upload. Set BUILD_EXPORT_ENCRYPTED_KEY to the number 1 to have Scriptoria generate an encrypted key. After you have uploaded the encrypted key to Google Play, you can remove this publishing property.

Share App Link

Publishing Property: BUILD_SHARE_APP_LINK

There are a couple of App Builders settings in the Features page for sharing. The first option is Share link to app on Google Play with the link based on the package name of the app. The third option is Share app installer file which allows sharing the actual app from phone to phone (e.g. using Bluetooth or Wifi-direct). Scripture App Builder defaults the first option unchecked and the third option checked (as seen below).

Navigation	Formatting	Annotations	History	Links	References	Search	Menu Items	Settings	Sharing	Editor
------------	------------	-------------	---------	-------	------------	--------	------------	----------	---------	--------

Sharing

Do you want to add a 'Share App' menu item in the app to make it easy for users to share the app with others?

☐ Share link to app on Google Play
Enable this option only if you have published your app on Google Play.

☐ Share link to app on website
Enable this option if you would like users to be able to share a link to download the app's APK file with other users.
Website URL

☒ Share app installer file
Enable this option if you would like users to be able to share the app's APK file with others.

☐ Share link to app on Apple App Store
Enable this option only if you have published your app on Apple App Store.

When Scriptoria was used only by SIL, it could only publish to Google Play so we decided it was best to always enable the Share link to app on Google Play option. This helps the maintainer of multiple apps so that this setting is always set and all the published apps to be consistent. If you would rather have Scriptoria use the setting in the App Builder project file, then set the value of the BUILD_SHARE_APP_LINK publish property to the number 0.

Publishing Property: BUILD_SHARE_DOWNLOAD_APP_LINK

For apps that are published to Google Play as an Android App Bundle (AAB), the Share app installer file feature is always disabled in the app (since the app delivered to the device will not work on all phones). The Share link to app on website feature was added so that apps that are distributed as an Android App Bundle can share a link to the full APK with other devices. If the app is being built on Scriptoria, you can set BUILD_SHARE_DOWNLOAD_APP_LINK to the number 1 and Scriptoria will enable the Share link to app on website feature and fill in the Website URL to a URL of the APK from the most recent successfully published app.

Publishing HTML and PWA

With Scripture App Builder, a project owner can generate HTML from a book collection that can be included in a website or used as a [Progressive Web App](#) (PWA) that can be installed on mobile and desktop devices. Scriptoria support publishing either HTML or PWA to a website using [rclone](#). Scriptoria stores in a secure area an rclone.conf config file which can define multiple remote locations. For each remote location it includes the credentials required to copy files to the cloud provider.

The rclone software supports publishing files to many different [cloud providers](#). The WebDAV protocol can be used for stand-alone websites. If the website uses Apache, the mod_dav module provides WebDAV functionality.

Use the rclone config command to create the configuration and test it. Here is an example:

```
[examplesite]
type = webdav
url = https://api.example.com
public_url = https://www.example.com
vendor = other
user = scriptoria
pass = dhXTY5BDI_36vIrhJgPjSGPRttKiBEw
```

To test that the rclone configuration works correctly, use the following commands:

```
rclone mkdir examplesite:dir
rclone copy . examplesite:dir
DATE=$(date +"%Y-%m-%d")
rclone mkdir examplesite:backup/dir/${DATE}
rclone copy example:dir example:backup/dir/${DATE}
```

If you have issues with the server-side copy (the last command), there may be additional

configuration that you need to perform if there is a proxy in front of your website.

Add the `public_url` property in the `rclone.conf` for the URL that is used by unauthenticated users to access the files. This will be used by Scriptoria to provide the publishing url in the Scriptoria UI.

The following publishing properties are specified in the SAB project.

Publishing Property: `PUBLISH_CLOUD_REMOTE`

Specify the remote (e.g. `examplesite` from the `rclone`) that should be used to publish the HTML or PWA. If not specified, then the first remote in the `rclone.conf` file will be used.

Publishing Property: `PUBLISH_CLOUD_REMOTE_PATH`

Specify the subdirectory of the remote where the files will be copied to. If not specified, then the files will be copied to the root directory of the remote or the home directory of the user used to connect to the server.

Add the `server_root` property in the `rclone.conf` file when the unauthenticated users access is not at the base of the remote path. For example:

```
[examplesite]
type = sftp
host = example.com
public_url = https://example.com
server_root = www
user = example
port = 22
key_pem = -----BEGIN RSA PRIVATE KEY----- ... -----END RSA PRIVATE KEY-----
use_insecure_cipher = false
```

If `PUBLISH_CLOUD_REMOTE_PATH = www/data/iso/pwa` and the web server root path is served from the `www` directory, the the correct public url would be <https://example.com/data/iso/pwa/index.html>. specifying `server_root = www` allows Scriptoria to build the correct public url for the published PWA.

Publishing Property: `BUILD_PWA_COLLECTION_ID`, `BUILD_HTML_COLLECTION_ID`

If there are multiple Book Collections in the project, specify the Book Collection Id (e.g. `C01`) of the collection to use for generating the PWA. By default, the first Book Collection is used.

You can publish multiple Book Collections to subdirectories of the remote path. Set the property with Book Collection ID followed by equal sign followed by subdirectory name. Then use a comma between each of these. Here is an example of 3 Book Collections.

`BUILD_HTML_COLLECTION_ID C01=web,C02=ulb,C03=udb`

Publishing Property: `PUBLISH_CLOUD_COMMAND`

Since Scriptoria uses [rclone](#) to publish files to cloud providers, there are two primary commands used to publish the files: sync or copy. Both of these use an intelligent transfer so that it doesn't attempt to copy unchanged files. The difference is that sync will delete files at the remote that don't exist at the source so that they match exactly after the publish and copy will only update files and leave other files alone. For PWAs, there are generated files where the filenames change so you should use sync. If not specified, the default value is sync.

Embedding Audio

Faith Comes By Hearing (FCBH) is one of the organizations who helped design the new version of Scriptoria so that they could use one project to build multiple apps for different distribution strategies:

- Expandable Edition - distribute the app over the internet (e.g. Google Play) with some of the audio embedded in the app and the rest downloaded from the internet
- Full Edition - distribute the app direct to the phone without internet (e.g. microSD cards or Wifi hotspot) with all of the audio embedded in the app

To enable building the same project with different distribution strategies, Scriptoria makes changes to the project configuration during the build that are not saved back to the project repository.

Download Audio During Build

To embed audio in an app, there is an Asset File Source so that files associated with this file source will be packaged inside the APK file. To do this, the audio has to be present during the build but we would prefer that the audio is not uploaded to the project repository. If the audio to be embedded is available in FCBH's Digital Bible Platform (DBP), then the audio can now be downloaded during the build. Additionally, the new version 4 of DBP has the MP3 audio available in multiple bitrates (16 or 64 Kbps are currently available for many languages). You will need a DBP4 key to download audio during the build.

Publishing Property: `BUILD_AUDIO_DOWNLOAD`

Set this to value of 1 if missing asset audio should be downloaded during the build. If not specified, the default value is 0.

Publishing Property: BUILD_AUDIO_DOWNLOAD_URL

Set this to a URL that implements the DBP4 API. If not specified, the default value is <https://4.dbt.io>.

Publishing Property: AUDIO_DOWNLOAD_MISSING_ASSETS_KEY

Set this to a dbt_key used to access the DBP4 API. A value must be specified. There is no default value.

Publishing Property: AUDIO_DOWNLOAD_BITRATE

Set this to the number of Kbps that should be downloaded during the build. If not specified, the default value is 64.

Publishing Property: AUDIO_DOWNLOAD_CODEC

Set this to the name of the codec that should be downloaded during the build. If not specified, the default value is mp3. Valid values are mp3 or opus.

Note: As of 4/26/2021, the valid combinations are: codec=opus/ bitrate=16 or codec=mp3/bitrate=64.

Example: FCBH Android Expandable Edition to Google Play

BUILD_AUDIO_DOWNLOAD=1

AUDIO_DOWNLOAD_BITRATE=16

AUDIO_DOWNLOAD_CODEC=opus

AUDIO_DOWNLOAD_MISSING_ASSET_KEY=<dbt_key>

Result:

For any audio file which has the source type=asset, if the audio is not included in the project, download the missing audio from DBP4. Determine the fileset from the filename and change it so that 16Kbps Ogg Opus files are downloaded.

Change Audio Source Before Build

Before building the app, you can change a FCBH file source to an Assert file source. When used with the previous settings, then the audio files used with the file source will be embedded in the app. Note that the source names are used.

Publishing Property: BUILD_AUDIO_UPDATE

Set this to value of 1 if audio sources should be updated before the build. If not specified, the default value is 0.

Publishing Property: AUDIO_UPDATE_SOURCE

Set this to <from_file_source>=<to_file_source> so that any file using the from_file_source is changed to use the to_file_source re(e.g. with AUDIO_UPDATE_SOURCE=DBP=Asset, files that were using the DBP file source are changed to use the Asset file source). Use the file source names. A value must be specified. There is no default value.

Example: FCBH Android Full Edition to S3

BUILD_AUDIO_DOWNLOAD=1

AUDIO_DOWNLOAD_BITRATE=16

AUDIO_DOWNLOAD_CODEC=opus

AUDIO_DOWNLOAD_MISSING_ASSET_KEY=<dbt_key>

BUILD_AUDIO_UPDATE=1

BUILD_AUDIO_UPDATE_SOURCE=DBP=Asset

Result:

For any audio file which has the source name of "DBP", change it to "Asset" (therefore all files that would have been downloaded now will be embedded).

For any audio file which has the source type=asset, if the audio is not included in the project, download the missing audio from DBP4. Determine the files set from the filename and change it so that 16Kbps Ogg Opus files are downloaded.