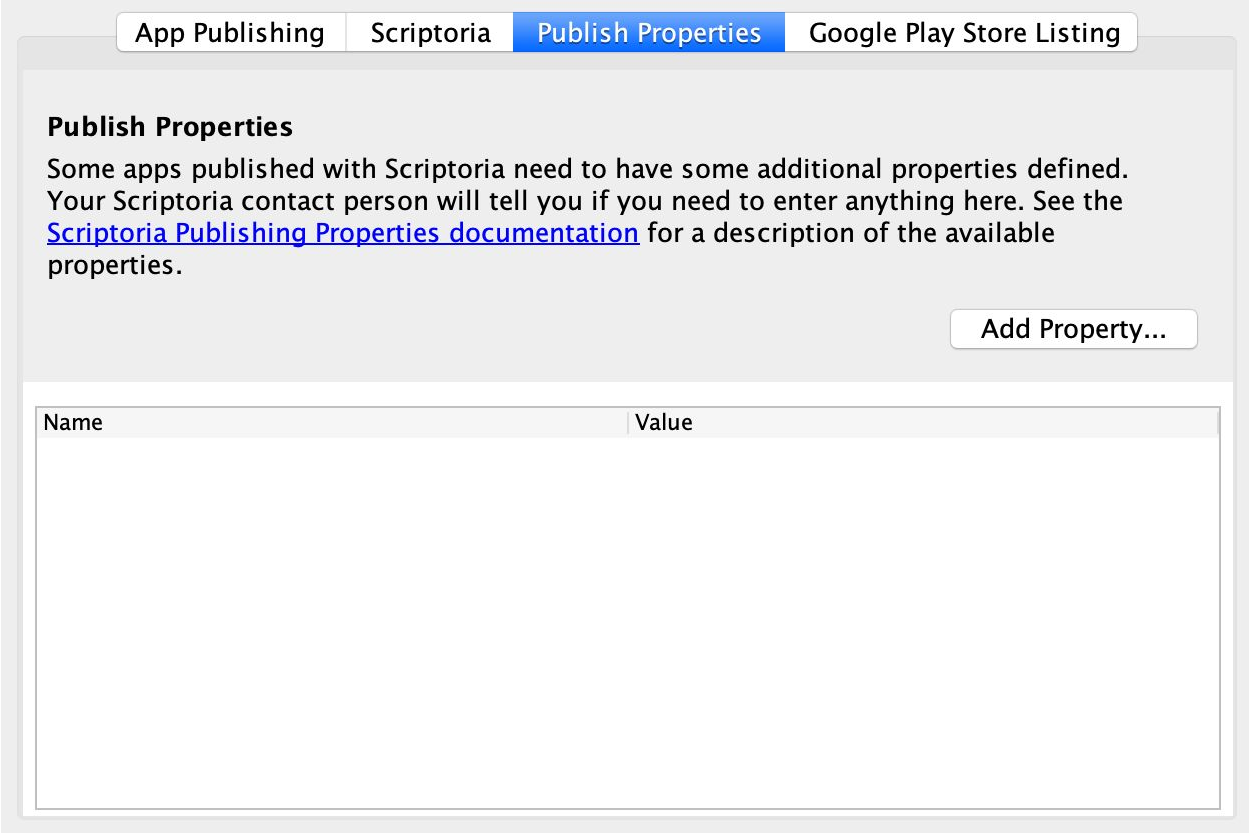
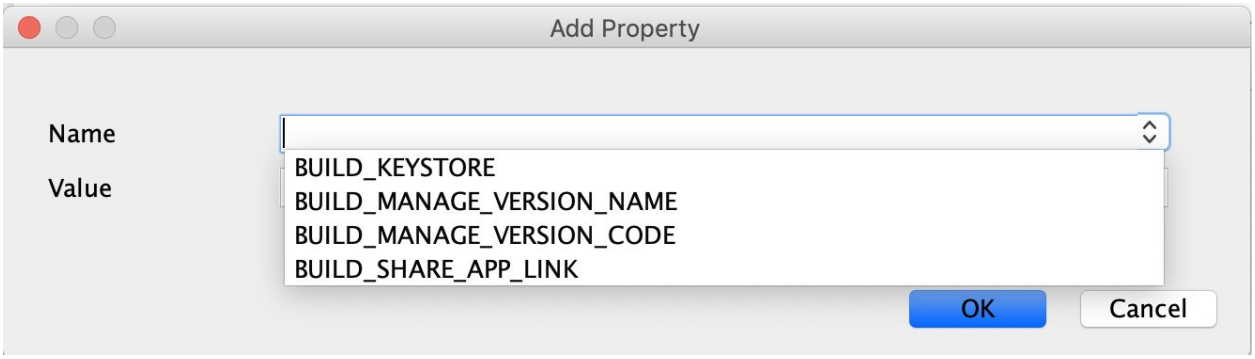
Scriptoria Publishing Properties

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.



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.



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

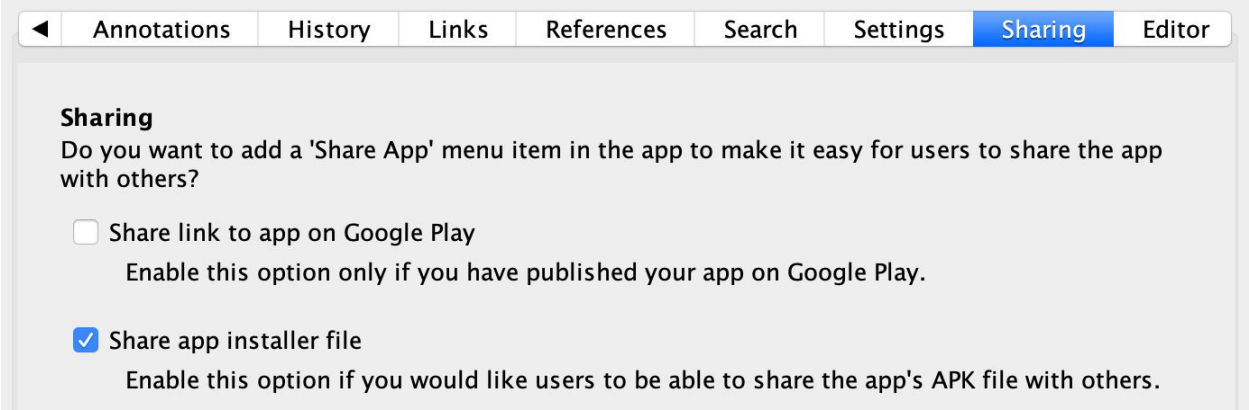
Publishing Property: `BUILD_MANAGE_VERSION_CODE`

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`.

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 second 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 second option checked (as seen below).



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 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`

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.

Backup Before Publishing

In developing this feature, it was requested that Scriptoria be able to backup the current files to a folder so that the files could be restored if there was an issue with the new files. There are two types of backups that can be performed: 1) server-side file copy to a backup directory, 2) download, compress to zip, and upload the zip to a backup directory. The backup directory will be a separate subdirectory on the server.

Publishing Property: `PUBLISH_CLOUD_BACKUP`

Set this to value of 1 if the files currently existing in the remote path should be copied to another location on the cloud provider before copying the new files. If not specified, the default value is 0.

Publishing Property: `PUBLISH_CLOUD_BACKUP_REMOTE_PATH`

Specify the subdirectory of the remote where the backup files will be copied to. If not specified, then the value defaults to "backups".

Publishing Property: `PUBLISH_CLOUD_BACKUP_ZIP`

Set this to the value of 1 if the backup files should be stored in a zip file. If not specified, then the value defaults to 0.

Examples

Given the configuration of examplesite above, here are a few scenarios and what the result would be:

Example 1:

```
PUBLISH_CLOUD_REMOTE=examplesite
PUBLISH_CLOUD_REMOTE_PATH=web
PUBLISH_CLOUD_BACKUP=1
```

Result:

copy of examplesite:web to examplesite:backups/web/2020-06-09_13-49-10
sync of new files to examplesite:web

Example 2:

```
PUBLISH_CLOUD_REMOTE=examplesite
PUBLISH_CLOUD_REMOTE_PATH=na/us/web
PUBLISH_CLOUD_BACKUP=1
PUBLISH_CLOUD_BACKUP_ZIP=1
```

Result:

copy of examplesite:na/us/web to build agent
zip of backup
copy of backup to examplesite:backups/na/us/web/web-2020-06-09_13-52-55.zip
sync of new files to examplesite:na/us/web