Wycliffe Associates
Media YAML Resource Downloader

Summer Hackathon 2020
8/11/2020 - 8/13/2020

Sponsor: Joe S.
Developer: David Simpson

Create a script to assemble assets for Resource Containers (Sponsor: Joe S)
NEED: Resource Containers need to be created on the fly building on the assets available in the Source Audio Server and WACS repos
DESCRIPTION: It is not practical to anticipate and store all the possible combination of source materials in resource containers that could be desired by users of Orature. Instead an app should poll the repos and source audio server to offer what is available for a given GL language and create the resource container on the fly. Then this app should bundle and deliver it to the user.

Resource Notes:

Good Example of a file format - includes version info.
Open Bible Stories - media.yaml

https://git.door43.org/unfoldingWord/en_obs/src/branch/master/media.yaml

Check with Christine - about the media.yaml files they create for: ULB

Resource Container Docs:

https://resource-container.readthedocs.io/en/latest/

This site is the official documentation for Door43’s Resource Container specification. Resource Containers (RCs) are the building blocks of content in the Door43 eco-system.

Need:

URL for translations.JSON file.
http://bibleineverylanguage.org/wp-content/themes/bb-theme-child/data/translations.json

Usage: This file will contain a list of all of the translations available, but doesn’t contain the info to derive GLs. Spreadsheet has this info now. FUTURE: In the future, this app could simply call the API which returns the list of GLs.

Need: Ask Christine

Need to know how to find the actual version info for each media block in order to replace the {latest} placeholder with the actual version number for downloading the media files.

Question:

Does the user need to select a subset of media files to download? Or all media?
For instance just one book of the Bible?
or just a selection of the media files found?

No - this is not required for the initial development. Might be a stretch goal.

Question:

Once the translation has been selected by the user, how does the app know the URL to use to get the manifest.yaml and additional files if needs?

Example:

https://content.bibletranslationtools.org/WA-Catalog/en_ulb/src/branch/master/manifest.yaml

https://content.bibletranslationtools.org/WA-Catalog/pt-br_ulb

Question:

What name should be used for the zip file created from the media files?

Idea from Joe: Append “-localcontent" to the original name.

Planning
Tasks:

1) outline of App - window, buttons, fields Temp Directory

2:37pm 8-11-2020

A) where will files get placed? Working directory location for instance

The list of GLs will be embedded for now.

2) ~~User does a Drag & Drop and gets~~ the list of GLs from translations.JSON

FUTURE- Use the API to get the list of GLs. Also use API to get top-level repository URL.

8-11-2020 2:37PM - Planned As Embedded For Now

The Translations menu gets populated with this list of values.

3) User Selects the GL they wait to work with. 8-12-2020 Use the language abbreviation for initial version.

4) Idea: Let user select a local working directory.

2:00PM 8-11-2020 Will use Local Temp Directory

Sub-folders will get created as needed

Zip file will get Created at the top level of This directory? or some other path for the zip file.

X 5) Q: Let user select list of files or books or chapters to download files from (?) Answer: Not required for v1.

6) App checks for existence of working folder - creates sub folders as needed.

Creates Zip file

Opens folder containing zip file.

Timestamp for Top-Level Enclosing Folder?

Planning Tasks :

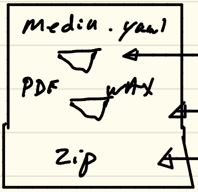
7) ULB – How to get URLs to repository ?

   A) User selects from a pick list of language IETF codes

   B) This info is used to construct the URL to the top-level of the repository.

Implementation Tasks:

✓ 1) Create new icon / graphic for app.
9:48AM 8-12-2020



2) Create outline of stack.
11:28AM 8-12-2020 Copy of previous stack - with changes.

New icon
No Drag & Drop icons
Language Menu
Repository Menu
Media URLs fields - Debug ?
URL field - Debug ?

3) Language Menu - Processing
2:18PM 8-12-2020 Construct a URL based upon the IETF language and the selected Repository.
Result: Gets written into URL field

4) Download button   Consider changing to: Get Media URLs

Processing: Phase #1

4:05PM 8-12-2020 Check for contents of URL field being empty. Error message.
Download manifest.yaml
Download media.yaml - if it exists
If media.yaml doesnt exist - put 'No media.yaml file' into first URL field
Clear 2nd URLs field

If media.yaml exists - create top-level directory inside Temp folder.

4) Download Button - Continued

Processing - Phase # 2

6:42PM 8-12-2020 → Find all of the media files - original URLs

7:45AM 8-13-2020 Create new versions of URLs - with place holders we mapped to use correct version # from top-level manifest.yaml file.

5) Directory Structure:

8:54AM 8-13-2020 Last part of URL

media

/identifier/ ← Media Block identifier (book)

filename.pdf

6) Zip File Creation   Consider: Download & Zip Files

~~Create top level Time stamp folder - in Temp~~ On:

9:53AM 8-13-2020 Create 2nd level folder - using top-level URL

Create directory structure for each file/folder as needed

10:23AM 8-13-2020 Download each file

Write each file info its folder

10:48AM 8-13-2020 Create zip file of 2nd level folder using:

folder name - local content

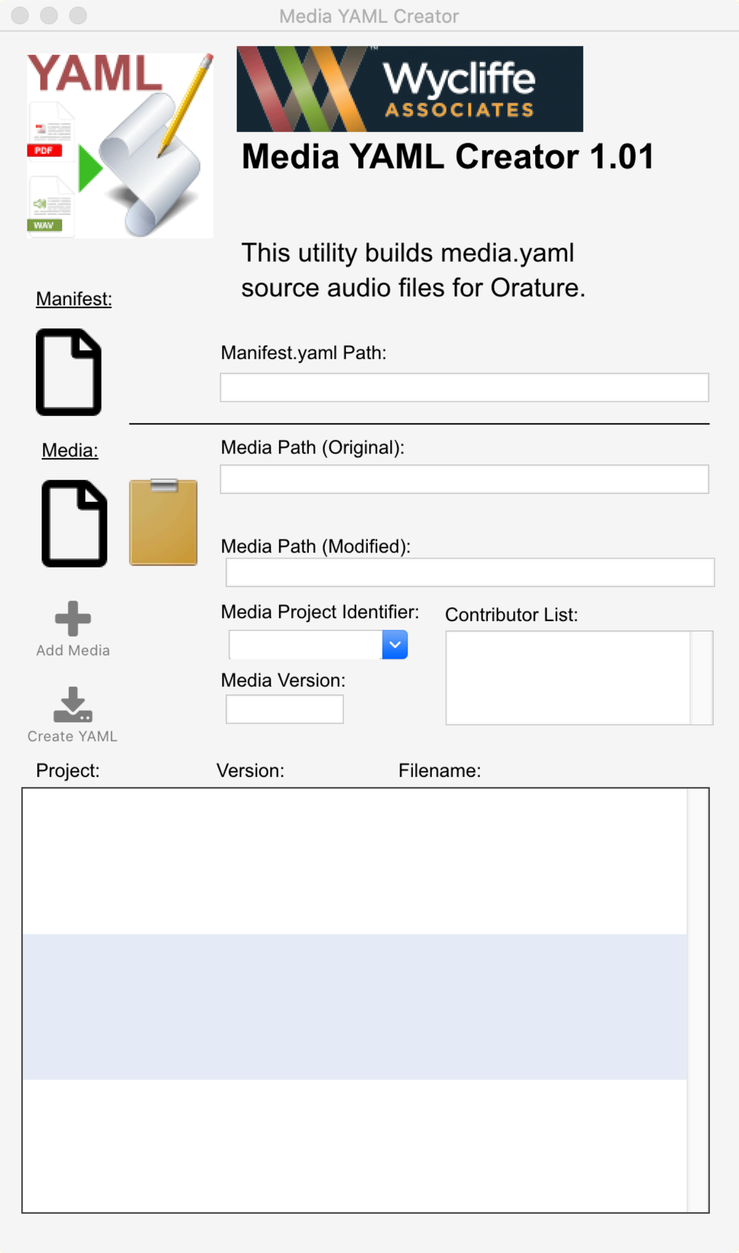
7) Add progress object - update during file processing

3:46PM 8-14-2020 Test with delay code - wait for 5 ticks

8) Fix - Add media.yaml & manifest.yaml to top level folder.

Fixed 11:35AM 8-14-2020

Previously Designed: Media YAML Creator App:

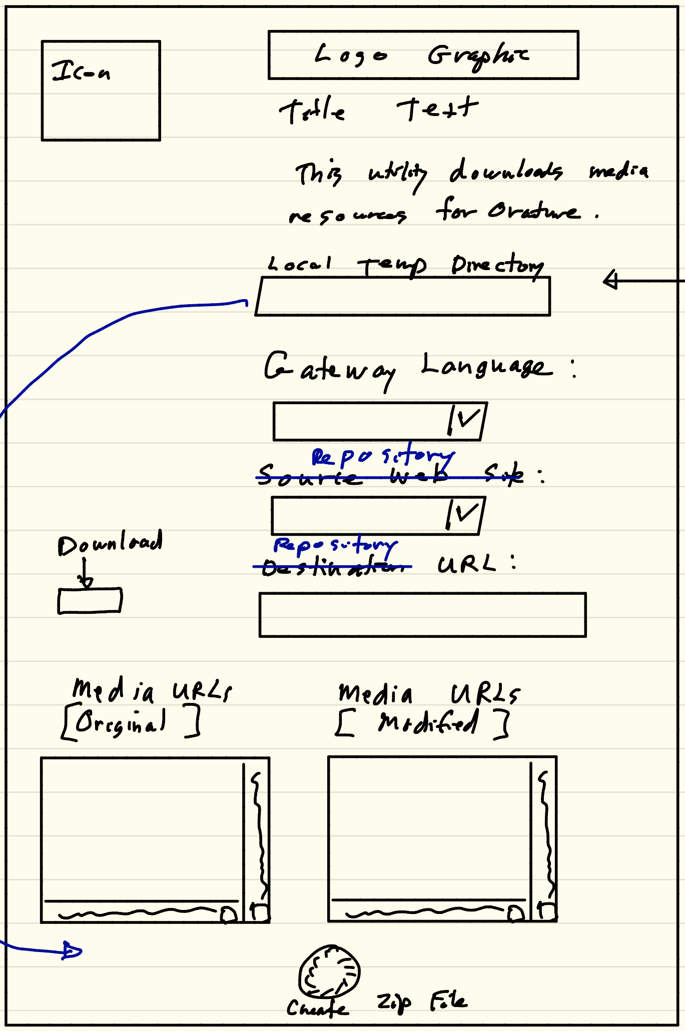


Media YAML Creator

Media YAML Creator 1.01

This utility builds media.yaml source audio files for Orature.

Media YAML Resource Downloader GUI



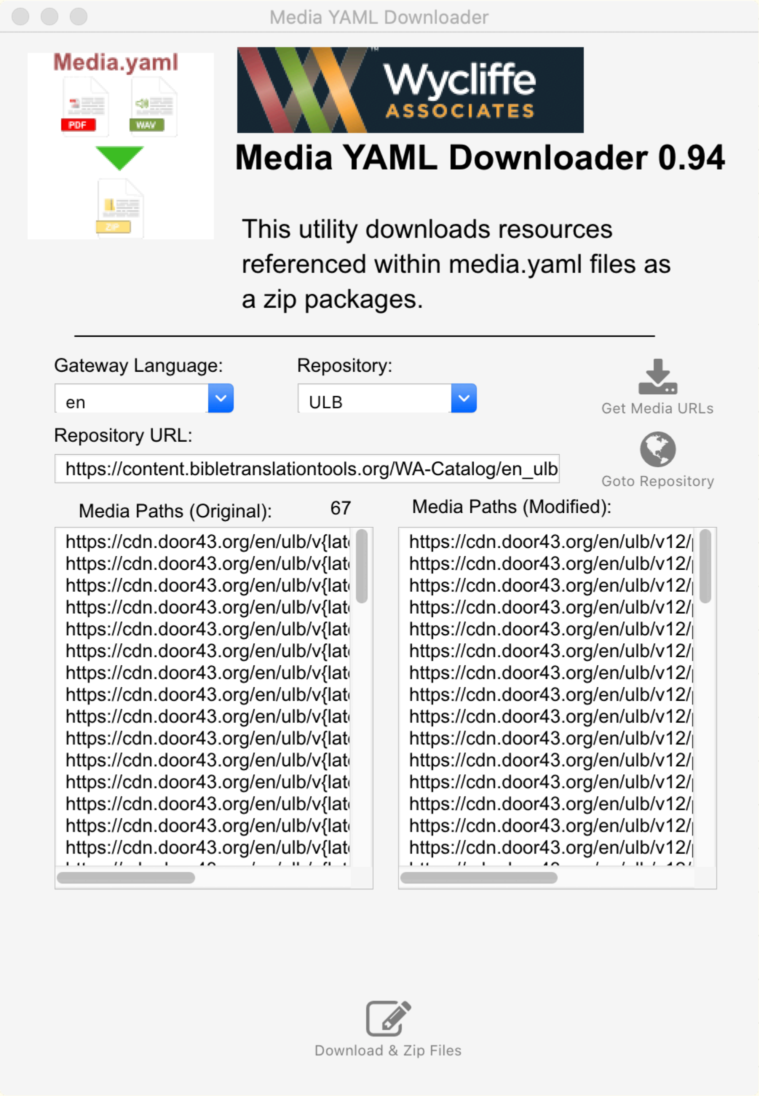
Not needed ?

GL Menu
[Embedded List for now

ULB ← version ?
OBS

Generated URL based upon selections

New: Media YAML Downloader App:



Media YAML Downloader 0.94

This utility downloads resources referenced within media.yaml files as a zip packages.

Gateway Language: en

Repository: ULB

Repository URL: https://content.bibletranslationtools.org/WA-Catalog/en_ulb

Media Paths (Original): 67

```
https://cdn.door43.org/en/ulb/v{lat
https://cdn.door43.org/en/ulb/v{lat
https://cdn.door43.org/en/ulb/v{lat
https://cdn.door43.org/en/ulb/v{lat
https://cdn.door43.org/en/ulb/v{lat
https://cdn.door43.org/en/ulb/v{lat
https://cdn.door43.org/en/ulb/v{lat
https://cdn.door43.org/en/ulb/v{lat
https://cdn.door43.org/en/ulb/v{lat
https://cdn.door43.org/en/ulb/v{lat
https://cdn.door43.org/en/ulb/v{lat
https://cdn.door43.org/en/ulb/v{lat
https://cdn.door43.org/en/ulb/v{lat
https://cdn.door43.org/en/ulb/v{lat
https://cdn.door43.org/en/ulb/v{lat
```

Media Paths (Modified):

```
https://cdn.door43.org/en/ulb/v12/
https://cdn.door43.org/en/ulb/v12/
https://cdn.door43.org/en/ulb/v12/
https://cdn.door43.org/en/ulb/v12/
https://cdn.door43.org/en/ulb/v12/
https://cdn.door43.org/en/ulb/v12/
https://cdn.door43.org/en/ulb/v12/
https://cdn.door43.org/en/ulb/v12/
https://cdn.door43.org/en/ulb/v12/
https://cdn.door43.org/en/ulb/v12/
https://cdn.door43.org/en/ulb/v12/
https://cdn.door43.org/en/ulb/v12/
https://cdn.door43.org/en/ulb/v12/
https://cdn.door43.org/en/ulb/v12/
https://cdn.door43.org/en/ulb/v12/
```

Example media.yaml (from Joe):

```
resource:
  version: '{latest}'
projects:
  -
    identifier: 'jas'
    version: '{latest}'
    media:
      -
       identifier: 'wav'
       version: '{latest}'
       contributor: []
       chapter_url: 'media/jas/
en_nt_ulb_jas_c0{chapter}.wav'


       chapter_url: -
       url: http:
```

Example manifest.yaml (from UnfoldingWord Site):

```
---
dublin_core:
  conformsto: 'rc0.2'
  contributor: []
  creator: 'Door43 World Missions Community'
  description: "An open-licensed update of the ASV, intended to provide a 'form-centric' understanding of the Bible. It
increases the translator's understanding of the lexical and grammatical composition of the underlying text by adhering
closely to the word order and structure of the originals."
  format: 'text/usfm'
  identifier: 'ulb'
  issued: '2017-11-29'
  language:
    direction: 'ltr'
    identifier: 'en-x-demo1'
    title: 'English'
  modified: '2017-11-29'
  publisher: 'unfoldingWord'
  relation:
    - 'en/tw'
    - 'en/tq'
    - 'en/tn'
  rights: 'CC BY-SA 4.0'
  source:
    -
      identifier: 'asv'
      language: 'en'
      version: '1901'
  subject: 'Bible'
  title: 'Unlocked Literal Bible'
  type: 'bundle'
  version: '12'
checking:
  checking_entity:
    - 'unfoldingWord'
  checking_level: '3'
projects:
  -
    title: 'James'
    versification: 'ufw'
    identifier: 'jas'
    sort: 60
    path: './60-JAS.usfm'
    categories: [ 'bible-nt'  ]
```

This identifier represents the entire manifest.yaml file.

This identifier info with be repeated for every project – There will be many projects contained within one manifest.yaml file.

FUTURE Enhancements:

The API for getting gateway languages could return the repository URL for ULB and OBS. Craig has mentioned that multiple “Primary” repos may be returned from this type of query.

Substitution of {quality} Placeholders:

Not enough testing has been done to determine if additional placeholders are actually being used in the published repository files.

Version key is displayed in at least 3 places in the YAML file format.

1) Top-level of manifest.yaml file. i.e. {12} or some number
2) Book level of the media.yaml file. i.e. {latest}
3) Media URL block where URL is defined. i.e. {latest}

Since the book level version is usually {latest} this version info is not currently being used by the app.

If the media.yaml URL block uses {latest} then the version number from the manifest.yaml file is used to construct the download URL.