Managing Metadata Overload: Automating E-Resource Workflows with Computer Scripts

Check E-Resource Access with the E-Resource Access Checker

What?
The E-Resource Access Checker is a Ruby script developed by Kristina Spurk to enable librarians to automate link-checking for electronic resources.

How?
The script checks a batch of titles for individual platforms. As long as the file you run the script on is a CSV (comma separated values) file, the script should work. The file does not need to be a KBART file or other format; you just need to ensure that the URLs are moved to the last column.

The script iterates through the CSV file, following each URL and reading the HTML on the page. It checks the HTML against a set of programmed conditions to evaluate if the publisher is providing access to the title. If access is not provided, the script will report that the title should be checked manually, and then the librarian can follow-up with the provider to resolve any access problems.

Github repository:
github.com/UMD-Libraries/Access-Checker

Kris Spurk’s article, “Getting What We Paid for: A Script to Verify Full Access to Electronic Resources”, https://journal.code4lib.org/article/9984

Create KBART files to supplement publisher data

What?
The MARCDownloader is a Python script that uses the WorldCat Search API to find and then transform MARC records into KBART.

- Used for: Improving discovery and access of resources by:
  - Supplementing knowledge base collections in WorldCat Collection Manager that are missing important data such as URIs or OCLC numbers
  - Creating collections when one is unavailable for subscription or open access platforms and collections.

How?
The script uses a query to search WorldCat, most often searching URLs to find a set of records for titles in a database or platform. It then iterates through the search results, converting the relevant MARC data into KBART. If desired, the script can be edited to clean the MARC21 XML data, especially for URLs in B56 fields. Searches should be targeted and concise because the API only provides access to the first 10,000 results. The script has a workaround to add title and author searching, but it is not effective and is a last resort. When completed, the output can be uploaded into Collection Manager or added to another collection as required.

Python WorldCat Search API

Under-development Github Repository: https://github.com/bosidy-benjamin/20W1RCsearch2MARC-wanderer

Collect Coverage Data and License Terms with KBQuery

What?
KBQuery is a Python script that runs automated batch searches, combining data from the WorldCat knowledgebase API and the WorldShare License Manager API. The script outputs the report as a tabled, separate Excel (XLSX) file.

- Used for:
  - Finding different sources your library has to access titles (electronic subscription, aggregator database, print, etc.)
  - Compiling coverage and licensing data to support collection development activities
  - Checking if entitlements are all selected in the knowledge base to support e-resource maintenance

How?
KBQuery reads a text file, created by the user, containing a list of search terms (ISBN, ISSN, OCLC Number or title). The script then runs a query against the knowledge base using each search term. If a match is found, it pulls out a set of data, including the available coverage, and writes it to a file. It also takes the Collection ID from the knowledge base API response to search for a corresponding license using the License Manager API. If a license is found, it searches for perpetual access rights an archival access rights and adds that to the output file.

The script has two primary modes:
1. Search within a single collection (ideal for maintenance work to ensure your entitlements are selected)
2. Search against the whole knowledge base (ideal for supporting collection development work by identifying your complete coverage).

WorldCat knowledgebase API
WorldShare License Manager API

Under-development Github Repository: https://github.com/bosidy-benjamin/20W1RSearch2XLSX