Library – Content & Access Services

Release Notes

April 2019 | Ver 0.5 (draft)

Alma Resource Sharing Partner update software

Installation notes

Murray Deerbon
PM / BA
La Trobe University
Victoria 3086

t: 03 9479 5065
e: m.deerbon@latrobe.edu.*au*
w: latrobe.edu.au

DOCUMENT CONTROL

Version	Date	Amendment	Distribution
0.1	11/01/2019	Initial draft	L. Brown, C. Ambrose
0.2	18/01/2019	Updated with feedback; changes to log and data file locations; configuration file changed to yaml format	
0.3	24/01/2019	Added disclaimer	
0.4	16/04/2019	Added revised code listing; revised config.yaml details. Revised location of config.yaml. Minor related edits and updates.	
0.5	30/04/2019	Changed program to use PUT instead of DELETE / POST; updated code listing	

Table of contents

1. OVERVIEW

Alma is being used for Inter Library Loans (aka Document Delivery). It has no inbuilt process to update partner information, i.e. those (Australian) Partner libraries that are active – or suspended – from lending services. To update this data manually is problematic, as the data changes quite frequently.

These notes describe a program written in Python 3.7, that automatically extracts the suspension data from the Libraries Australia website and updates Alma accordingly. The program retrieves (screen scapes) from the Libraries Australia website all the Australian Libraries that are listed including their status of active or suspended and updates the Alma record as appropriate via the Alma REST API.

Python was chosen to do this task as it is free, has an expansive library of extensions and can be deployed on all the major desktop platforms (i.e. Windows, Apple Mac & Linux). This solution doesn't require server technology, databases or other specialist requirements.

Note: *A more comprehensive solution prepared by Macquarie University is offered here: https://github.com/mqlibrary/resource-sharing-partners-sync/blob/master/docs/README.md which includes the maintenance of the New Zealand Library data as well.*

1.1. Process description

When the program is invoked, it first screen scrapes the data from Libraries Australia and stores the results in a CSV file. It then starts to update Alma with the partner information saved to the CSV file. Only those partner libraries that have a status change are updated. If the status of a library goes from being Active to Inactive (or on the Libraries Australia website from 'Suspended' to 'Not suspended'), the record is extracted from Alma, the status is changed, the Alma record is updated (PUT). The log file is updated with the transaction.

When all the records in the CSV file are processed, the program terminates, after closing all the files, updating the log files and the user interface.

2. SUPPLIED FILES

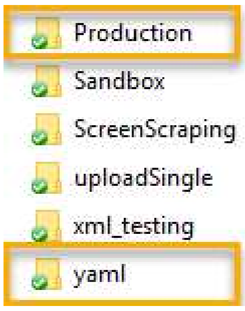
2.1. Package

In the software package there are five files:

Filename	Description
AlmaPartnerUpdate_V1.xx.py	The python code
SampleConfig.yaml	Contains the Libraries Australia members and suspension web page URL, the URL of API and your unique API key. This file needs to be renamed config.yaml, and placed in a sibling folder named yaml.
ReleaseNotesV0.5.docx	This document

Filename	Description
GetAll_Alma_PartnerRecords_v0.3.py	A program to extract in XML format, all the partner records from Alma, 100 records at a time.
Readme.txt	A read me file with simple description and instructions

To use the Alma Partner update requires Python 3.7 to be installed on the machine. A **SampleConfig.yaml** file is provided and can be edited to suit local requirements. It should be renamed to **config.yaml** and be installed in a `yaml` folder at the same level (i.e. a sibling folder) as the program file.



2.2. Software

Install Python. Go to https://www.python.org/downloads/release/python-371/ to download the software for installation.

For instructions on how to install can be found here:
https://wiki.python.org/moin/BeginnersGuide/Download
or here:
https://automatetheboringstuff.com/chapter0/ and search for 'Downloading and Installing Python'.

The program also requires python add-ins that are not part of the initial installation of Python 3.7. They include beautifulsoup4, lxml, requests, pyyaml and lxml.

Use the PIP command to install these packages. For more information on `PIP`, see https://www.w3schools.com/python/python_pip.asp.

2.3. Input files

When the software runs it reads a configuration file ('..`\yaml\`**config.yaml**') that contains data specific to your installation. The configuration conforms to the 'yaml' specification, in the form '`variable_name : value`'.

Name	(Sample) Value	Purpose
ladd_url	https://www.nla.gov.au/librariesaustralia/connect/find-library/ladd-members-and-suspensions	Libraries Australia members and suspension web page URL
APIKey	l7xxx99dd9db9x7646x0x377d914585x25x4	Contains the unique API key to your installation
sand_APIKey	l8xxf199be9xxx894267xxx2f69zz99cbfb9	Contains the unique API key to your sandbox installation

Name	(Sample) Value	Purpose
alma_url	https://api-eu.hosted.exlibrisgroup.com/almaws/v1/partners	Contains the URL of your instance of Alma
sand_alma_url	https://api-eu.hosted.exlibrisgroup.com/almaws/v1/partners	Contains the URL of the sandbox instance of Alma
numrecs	1197	Used by the GetAll_Alma_PartnerRecords_v0.3.py
Sandbox	True	or False

The value of the 'ladd_url' (above) will not need to be changed, unless Libraries Australia change their URL.

The 'APIKey' is unique to your installation and will have to be updated accordingly. The key in the example above (and provided in the included yaml file) is fictional.

Similarly, the 'sand_APIKey' is unique to your installation and will have to be updated accordingly. The key in the example above (and provided in the included yaml file) is fictional.

The value of the 'alma_url' is the url of the Alma installation where the webservices reside.

The value of the 'sand_alma_url' is the url of the Alma installation where the webservices reside.

To create or edit the yaml file, use a simple text editor to change the data (e.g. `notepad.exe` or similar); not MS Word.

This file must be installed in a sibling folder (yaml) as python program; **Alma_PartnerUpdate_V1.06.py**.

2.4. Output files

When the program (**Alma_PartnerUpdate_V1.06.py**) runs, it creates a CSV data file and both updates and creates log files.

2.4.1. Data file

The first part of the program extracts ('screen scrapes') data records from a webpage on the Libraries Australia website (as specified by the **`ladd_url`**) and outputs this record to a CSV file. The record contains the Libraries Australia member identifier (NUC[1] code), and their suspension status. The output filename is of the form '**LibAusSuspendlstyyyymmdd_HHMMSS.CSV**; e.g. **LibAusSuspendlst20190110_151231.CSV**. This file is saved to the '**`.\data`**' folder.

A new CSV file is created each time the program runs. The most recently created CSV file is used to update Alma and is retained for information purposes only. It can be deleted at any time after the update is completed.

2.4.2. Log files

The program also updates a 'harvest' log file **rotaScreenScrape.log** with the date and file name of the most recently created CSV file, and a count of the number of records extracted from the

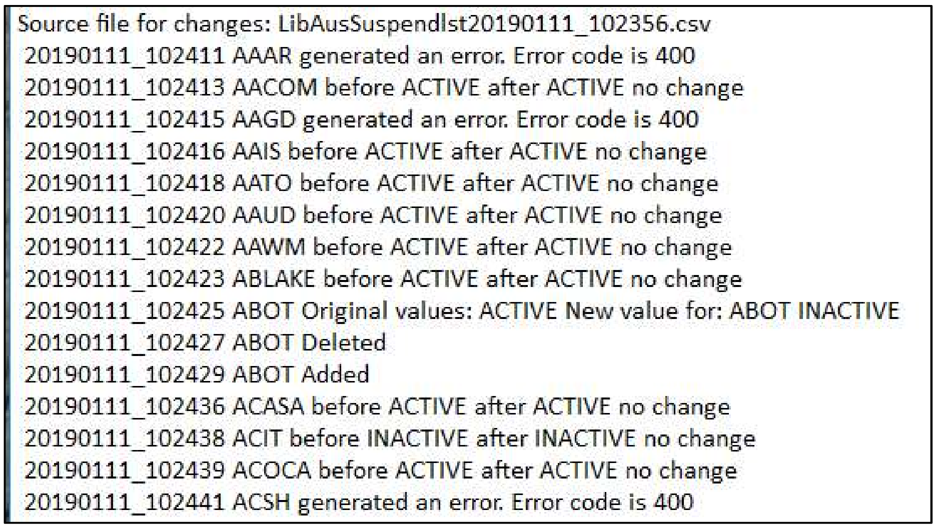
[1] NUC: National Union Catalogue, a unique code assigned to Libraries that make their resources available for loan.

website. This file grows and can be deleted if it gets too large; the program automatically re-creates it if it is not there.

Once the extract of the data from the website has completed, the program starts to update Alma with the extracted data. A new logfile is created of the form **logfileyyyymmdd_HHMMSS.log**, e.g. **logfile20190111_102356.log**. The log files are saved to the '`.\logs`' folder.

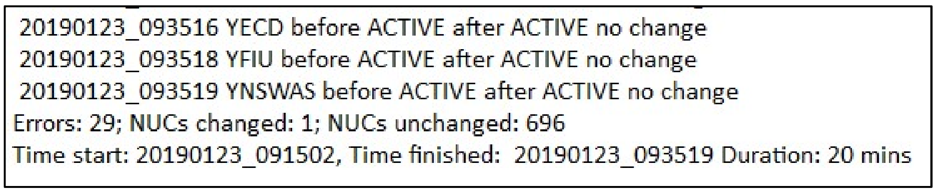
The first line records the input file (e.g. **LibAusSuspendlst20190110_151231.CSV**). It then lists all the NUC codes that were accessed from the suspension file, and the action taken in Alma. Each line has the date and time together with the NUC and the status or action taken.

The last line of the file is a summary listing the number of errors e.g. NUC's not found, NUC's changed and NUC's unchanged, as well as the date-time started, and the date-time finished, and the duration. The update takes about 7 minutes – updating 726 records in the production environment; up to 21 minutes in a sandbox.



```
Source file for changes: LibAusSuspendlst20190111_102356.csv
 20190111_102411 AAAR generated an error. Error code is 400
 20190111_102413 AACOM before ACTIVE after ACTIVE no change
 20190111_102415 AAGD generated an error. Error code is 400
 20190111_102416 AAIS before ACTIVE after ACTIVE no change
 20190111_102418 AATO before ACTIVE after ACTIVE no change
 20190111_102420 AAUD before ACTIVE after ACTIVE no change
 20190111_102422 AAWM before ACTIVE after ACTIVE no change
 20190111_102423 ABLAKE before ACTIVE after ACTIVE no change
 20190111_102425 ABOT Original values: ACTIVE New value for: ABOT INACTIVE
 20190111_102427 ABOT Deleted
 20190111_102429 ABOT Added
 20190111_102436 ACASA before ACTIVE after ACTIVE no change
 20190111_102438 ACIT before INACTIVE after INACTIVE no change
 20190111_102439 ACOCA before ACTIVE after ACTIVE no change
 20190111_102441 ACSH generated an error. Error code is 400
```

Figure 1 – Sample log file output – header section



```
 20190123_093516 YECD before ACTIVE after ACTIVE no change
 20190123_093518 YFIU before ACTIVE after ACTIVE no change
 20190123_093519 YNSWAS before ACTIVE after ACTIVE no change
Errors: 29; NUCs changed: 1; NUCs unchanged: 696
Time start: 20190123_091502, Time finished:  20190123_093519 Duration: 20 mins
```

Figure 2 – Sample log file output – tail section

Once the program has successfully completed, the log files and / or CSV files can be archived or deleted without ongoing impact to subsequent execution. *Do not delete the '**config.yaml**' file.*

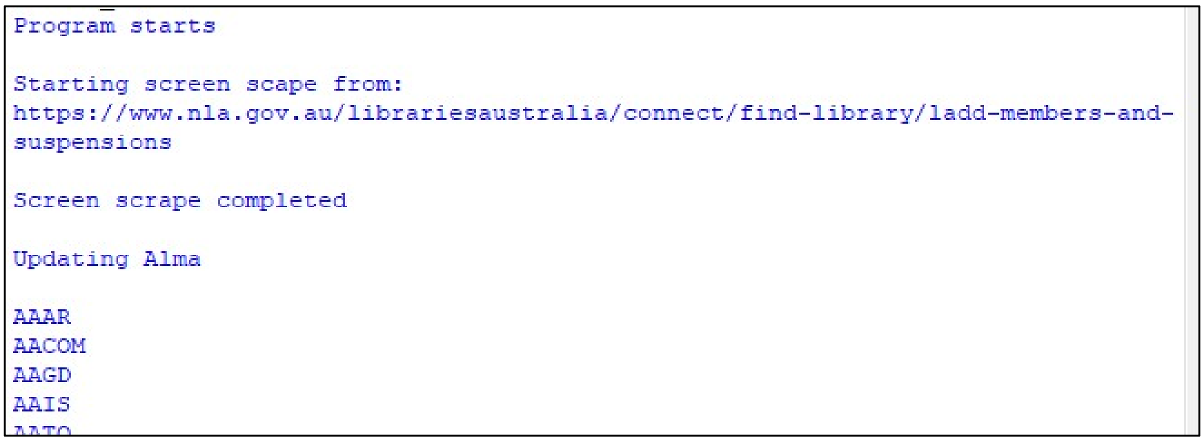
2.5. Screen output

When the program runs, the Python shell opens, and the program writes to the output screen the current NUC code being checked. This window can be minimised (not closed) without interruption to the program.

The displayed output in the Python shell identifies the various stages of the process; e.g. "Program start", "Starting screen scrape from:" {url}; "Screen scrape completed", "Updating Alma" each NUC as it is being processed. On completion, the program displays some statistics, and the message "Program finished".

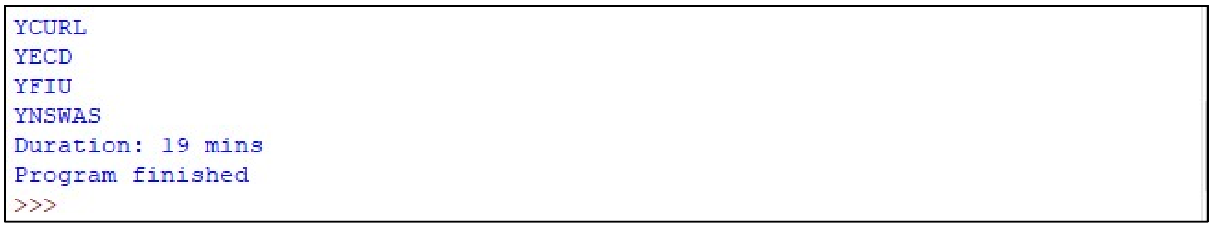
This is a batch process; there is no user input required as the program runs. The following are sample screen shots of the output the program produces as it runs.

If the program is invoked automatically (e.g. using Windows task schedular or cron task) the Python shell automatically closes at program completion.



```
Program starts

Starting screen scape from:
https://www.nla.gov.au/librariesaustralia/connect/find-library/ladd-members-and-
suspensions

Screen scrape completed

Updating Alma

AAAR
AACOM
AAGD
AAIS
AATO
```

Figure 3 - Commencing screen output



```
YCURL
YECD
YFIU
YNSWAS
Duration: 19 mins
Program finished
>>>
```

Figure 4 - Completion message

3. CODE

3.1. Python code

```
import requests
from urllib.parse import urlencode, quote_plus
from urllib.request import Request
from urllib.request import urlopen as uReq
from bs4 import BeautifulSoup as bs
from datetime import datetime
import time
from http import HTTPStatus
import os
import yaml
import sys
import ssl
ssl._create_default_https_context = ssl._create_unverified_context

ts = datetime.now()  # time start
## ######################################################################
## Written by Murray Deerbon - La Trobe University - January 2019
## mdeerbon@latrobe.edu.au
##
## Program to screen scrape 'LADD members & suspensions' website.
## Results are stored in a CSV file
## CSV file results are then used to update the Alma Partner records
## marking partners as Active or Inactive, as appropriate.

## Version 1.06 April 2019
## Updated to change to a PUT rather than a Delete & Post
## Included the SSL import
## #######################################################################

def check_for_code(a_key, cde, url):  # checks for a partner, and if it exists, returns the XML
    url =  url +'/{partner_code}'.replace('{partner_code}',quote_plus(cde))
    queryParams = '?' + urlencode({ quote_plus('APIKey') : a_key  })
    r = requests.get(url + queryParams)
    return r.status_code, r.text

def put_code(a_key, cde, values, url):  # PUTS the revised record
    url =  url +'/{partner_code}'.replace('{partner_code}',quote_plus(cde))
    queryParams = '?' + urlencode({ quote_plus('APIKey') : a_key  })
    values = values.encode()
    headers = {  'Content-Type':'application/xml'  }
    request = Request(url + queryParams
                      , data=values
                      , headers=headers)
    # PUT the data
    request.get_method = lambda: 'PUT'
    # read the data that has just been PUT
    response_body = uReq(request).read()  ## changed urlopen to uReq
    return

def now():
    dt = datetime.now().strftime(' %Y%m%d_%H%M%S ')
    return dt

# create folders for data and log files
dirLog='./logs'
dirData='./data'
yaml_folder = '../yaml/'
config = yaml_folder + 'config.yaml'

for d in [ dirLog, dirData ]:
    if not os.path.exists(d):
        os.makedirs(d)  #  create folders if they don't exist

# load config data
try:
    with open(config, 'r') as f:
        conf = yaml.load(f, Loader=yaml.FullLoader)
except FileNotFoundError:
    print(f'{config} file not found')
    print('This file is critical and the program will not run without it.')
    print('Press cancel at the window pop-up')
    quit()   # program terminates here if config file not found
```

```
except:
    print('Unexpected error: ', sys.exec_info()[0])
    quit()

print('Program starts\n')
print('Starting screen scape from:')

## define counters and other variables
log_string = ""
del_flag = False
newline = ' \n'
changed = 0
unchanged = 0
errors = 0
logtime = now().strip()
start_time = now().strip()
logfileName = dirLog + '//' + 'logfile' + logtime + '.log'  # log file for Alma Partner update

#####Screen Scrape variables #######
log_file_ss = dirLog + '//' + 'rotaScreenScrape.log'    # log file for screen scrape
ladd_url = conf['ladd_url']
print(ladd_url,'\n')

csv_in = dirData + '/' + 'LibAusSuspendlst' + now().strip() + '.csv'  ## CSV  file that holds the LADD
data

### get web site page ###
data = uReq(ladd_url)
page_html = data.read()
data.close()

#####################################################################################

## get config.yaml data
url = conf['alma_url']
s_url = conf['sand_alma_url']
APIKey = conf['APIKey']
s_APIKey = conf['sand_APIKey']
sandbox = conf['sandbox']
#####################################################################################

if sandbox == True:
    url = s_url
    APIKey =s_APIKey
    print(url)
    print(APIKey)
    print('Updating ALMA Sandbox')
    logfileName = logfileName + '_sand' + '.log'
    csv_in = dirData + '/' + 'LibAusSuspendlst' + now().strip() + '_sand' + '.csv'  ## CSV  file that
holds the LADD data
    in_key = input('Press the any key to continue: ')
else:
    print(url)
    print(APIKey)
    print('Updating ALMA Production')
    csv_in = dirData + '/' + 'LibAusSuspendlst' + now().strip() + '.csv'  ## CSV  file that holds the
LADD data
    #in_key = input('Press the any key to continue: ')

#######################################################################################

# html parsing
page_soup = bs(page_html, 'html.parser')
# grabs all the suspension directory table
table_content = page_soup.find("table", {"class":"views-table cols-6"})
tbody = table_content.find('tbody')
count_rec = 0
with open(csv_in,'w') as f:
    headers = 'nuc,lib_name,iso_ill,suspended,date_from,date_to \n'
    f.write(headers)
    for tr in tbody.findAll('tr'):
        nuc = tr.findAll('td')[0].text.strip()
        lib_name = tr.findAll('td')[1].text.strip()
        iso_ill = tr.findAll('td')[2].text.strip()
        suspended = tr.findAll('td')[3].text.strip()
        date_from = tr.findAll('td')[4].text.strip()
        date_to = tr.findAll('td')[5].text.strip()
        string = (nuc +','+ lib_name.replace(',',' ') +','+ iso_ill +','+ suspended +','+
date_from.replace(',',' ')  + ','+ date_to.replace(',',' ') +"\n").strip()
        f.write(string + '\n')
        count_rec += 1
```

```
with open(log_file_ss,'a') as lg:
    lg.write(f'Date time: {now()} filename: {csv_in}, records {count_rec} \n')

### - the CSV file of the rota changes has been created and the log file updated
print('Screen scrape completed\n')
print('Updating Alma\n')

with open(logfileName, 'w') as lg:
    lg.write(f'Source file for changes: {csv_in}\n')
    with open(csv_in,'r') as fin:   # opens the file with all the partner codes
        next(fin)                   # skips the first line to ignore the headers
        for lines in fin:
            line = lines.split(',')
            nuc = line[0].upper()
            suspend = line[3]
            #print(nuc)
            if suspend == 'Suspended':
                suspend = 'INACTIVE'
            else:
                suspend = 'ACTIVE'
            new_values = 'New value for: ' + nuc + ' ' + suspend  # comment for log file
            valid_code = check_for_code(APIKey, nuc, url)  # uses internal get function
            if valid_code[0] == HTTPStatus.OK:
                soup = bs(valid_code[1], 'xml') # covert to a 'soup' object
                active_stat = soup.find('status')
                a_s = active_stat.string    # the string component of the (active status)
                if a_s == suspend:          # there is no need to delete and re-post
                    log_string = now() + nuc + ' before ' + suspend + ' after ' + a_s + ' no change ' + 
newline  #####
                    lg.write(log_string)
                    unchanged += 1
                else:
                    active_stat.string = suspend  # the new value is assigned here
                    log_string = now() + nuc + ' Original values: ' + a_s + ' ' + new_values + newline
                    lg.write(log_string)
                    print(log_string)
                    put_partner = put_code(APIKey, nuc, soup, url)
                    changed += 1
            else:
                valid_code[0] != HTTPStatus.OK   # 200
                log_string = now() + nuc + " generated an error. Error code is " + str(valid_code[0]) + 
newline
                lg.write(log_string)        # log file update

                errors += 1
    lg.write(f'Errors: {errors}; NUCs changed: {changed}; NUCs unchanged: {unchanged} \n')
    lg.write(f'Time start: {start_time}, Time finished: '+ now())
### duration calcuations ########
    tstop = datetime.now() # time stop
    diff = tstop - ts
    mins = int((diff.total_seconds()) / 60 )
    secs = int(diff.total_seconds() - (mins * 60))
    lg.write(f'Duration: {mins} minutes {secs} seconds')

print(f'Duration: {mins} minutes {secs} seconds')
print(f'Errors: {errors}; NUCs changed: {changed}; NUCs unchanged: {unchanged}')
print('Program finished')
```

3.2. Config file

Save the following in the file **config.yaml**. Ensure that this file is in a yaml folder at the same (i.e. sibling) level as that holding the python code above.

```
ladd_url : https://www.nla.gov.au/librariesaustralia/connect/find-library/ladd-
members-and-suspensions
sand_alma_url : https://api-eu.hosted.exlibrisgroup.com/almaws/v1/partners
alma_url : https://api-ap.hosted.exlibrisgroup.com/almaws/v1/partners
sand_APIKey : l7xxx99dd9db9x7646x0x377d914585x25x4
APIKey : l8xxx99dd9db9x7646x0x377d914585cbfb1
numrecs : 1197
```

Note: *the APIKeys listed here are fictional.*

~ End of document ~