Building Web Apps to Streamline FOLIO Workflows

Guy Dobson, Drew University
gdobson@drew.edu
see also folio.drew.edu

Why would you want to?

to streamline FOLIO workflows

+ to do things that the UI isn't prepared to do

= to make life easier for you and your colleagues

backstory, basics, caveats, and disclaimers

backstory

Approx 20 years ago, when I was working for the Bergen County Cooperative Library System, I took Sirsi's week-long API course.

When I started working at Drew, on 2/14/2011, 20% of my time was devoted to implementing Kuale Ole with 4 other NJ academic libraries.

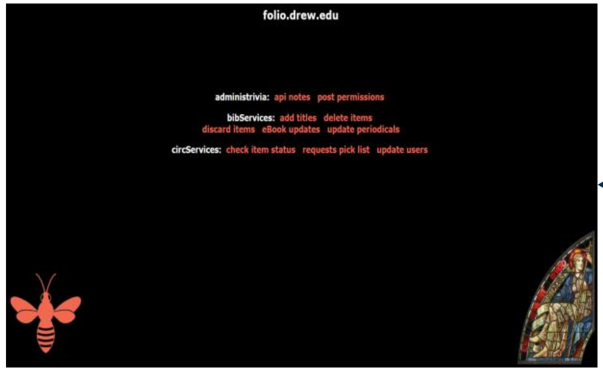
In 2019 Drew University was looking to move self-hosted services, like our Sirsi Symphony ILS, to the cloud.

In 2020 we signed a 5-year contract with EBSCO to host a FOLIO environment for us w/out an implementation team.

Since we were self-hosted, and I was familiar with Sirsi's API, I was able to get our data out of Symphony and load it into FOLIO.

basics : how do web apps work?

librarian



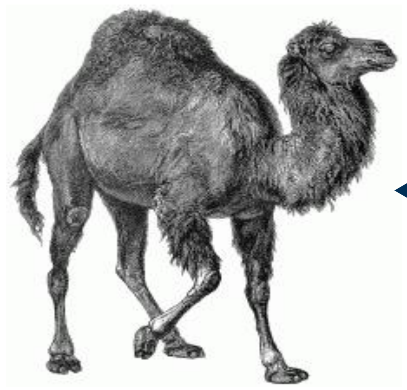
folio.drew.edu

administrivia: api notes post permissions

bibServices: add titles delete items
discard items eBook updates update periodicals

circServices: check item status requests pick list update users

web app



perl

drew.folio.ebsco.com

folio.drew.edu

EBSCO USER GROUP

basics : how do APIs work?

curl {what} {who} {where}

{what} : POST (create), GET (read), PUT (update), or DELETE

{who} : an X-Okapi-Token, included as a header*, that is user specific

*the headers also include Content-type and X-Okapi_Tenant information

{where} : a task specific endpoint that may or may not end with a query or a UUID

when {what} requires data it is supplied in the form of a JSON document as is any data that is returned (though not all endpoints have anything to say when they are done – when this happens no news is good news, usually) – e.g…

curl -s -X GET -H $headers https://okapi-drew.folio.ebsco.com/instance-storage/instances?query=hrid==in00001095127

$$\texttt{curl -s -X GET -H \$headers https:/}\underbrace{\texttt{/okapi-drew.folio.ebsco.com}}_{\text{base URL}}\underbrace{\texttt{/instance-storage/instances}}_{\text{endpoint}}\texttt{?query=hrid==in00001095127}$$

…returns:

basics : how do APIs work?

{

"instances": [

{"id":"371a1dba-85e8-4fc1-99ac-259938ca2ed5","_version":2,**"hrid":"in00001095127"**,"source":"MARC",**"title":"There's treasure everywhere / a Calvin and Hobbes collection by Bill Watterson."**,"indexTitle":"There's treasure everywhere /","alternativeTitles":[{"alternativeTitleTypeId":"8a3b90b0-c9e9-4e0e-b921-1aed031f9fc6","alternativeTitle":"Calvin and Hobbes. Selections"},{"alternativeTitleTypeId":"fe19bae4-da28-472b-be90-d442e2428ead","alternativeTitle":"There is treasure everywhere"}],"editions":[],"series":["Calvin and Hobbes."],"identifiers":[{"value":"95083102","identifierTypeId":"4d9a5d3a-cb8d-4783-8e5b-65df5d09c15d"},{"value":"(ckey)a944382","identifierTypeId":"fe19bae4-da28-472b-be90-d442e2428ead"},{"value":"(Sirsi) a944382","identifierTypeId":"fe19bae4-da28-472b-be90-d442e2428ead"},{"value":"(OCoLC)ocm34360465","identifierTypeId":"fa02ef8b-b0be-4183-ad46-38060d23c402"},{"value":"0836213130","identifierTypeId":"fabf0120-1511-4dee-b5c5-8e23da43ea84"},{"value":"9780836213133","identifierTypeId":"fabf0120-1511-4dee-b5c5-8e23da43ea84"},{"value":"0836213122 (paperback)","identifierTypeId":"fabf0120-1511-4dee-b5c5-8e23da43ea84"},{"value":"9780836213126 (paperback)","identifierTypeId":"fabf0120-1511-4dee-b5c5-8e23da43ea84"},{"value":"9780740777950","identifierTypeId":"fabf0120-1511-4dee-b5c5-8e23da43ea84"},{"value":"9781417775842 (turtleback books)","identifierTypeId":"fabf0120-1511-4dee-b5c5-8e23da43ea84"},{"value":"0740777955","identifierTypeId":"fabf0120-1511-4dee-b5c5-8e23da43ea84"},{"value":"141777584X","identifierTypeId":"fabf0120-1511-4dee-b5c5-8e23da43ea84"},{"value":"9781415505199 (Paw Prints)","identifierTypeId":"fabf0120-1511-4dee-b5c5-8e23da43ea84"},{"value":"1415505195 (Paw Prints)","identifierTypeId":"fabf0120-1511-4dee-b5c5-8e23da43ea84"},{"value":"9780590972086","identifierTypeId":"fabf0120-1511-4dee-b5c5-8e23da43ea84"},{"value":"0590972081","identifierTypeId":"fabf0120-1511-4dee-b5c5-8e23da43ea84"},{"value":"(OCoLC)34360465","identifierTypeId":"fa02ef8b-b0be-4183-ad46-38060d23c402"}],"contributors":[{"name":"Watterson, Bill,","contributorTypeText":"author, artist.","contributorNameTypeId":"9c97ac45-f339-45db-a0a7-2bd33309adb9","primary":true},{"name":"Harry A. Chesler Collection of Cartoon Art and Graphic Satire","contributorNameTypeId":"542a89f1-33a0-4072-9899-f6297e8d0709","primary":false}],"subjects":["Calvin and Hobbes (Comic strip)","Calvin (Fictitious character : Watterson)--Comic books, strips, etc","Hobbes (Fictitious character)--Comic books, strips, etc","Wit and humor, Pictorial","Comic books, strips, etc","Comics (Graphic works)"],"classifications":[{"classificationNumber":"PN6728.C34 W3874 1996","classificationTypeId":"fe19bae4-da28-472b-be90-d442e2428ead"},{"classificationNumber":"741.56973","classificationTypeId":"fe19bae4-da28-472b-be90-d442e2428ead"}],"publication":[{"publisher":"Andrews and McMeel","place":"Kansas City, Missouri", "dateOfPublication":"[1996]","role":"Publication"},{"dateOfPublication":"♭1996"}], "publicationFrequency":[], "publicationRange":[], "electronicAccess":[], "instanceTypeId":"03410cfb-1fca-4ea4-9cce-b03cd0477d03","instanceFormatIds":["3d68d5eb-8ca8-4ca0-81c1-a07c3abdca9d"],"instanceFormats":[],"physicalDescriptions":["175 pages : chiefly illustrations (some color) ; 23 x 31 cm."],"languages":["eng"],"notes":[{"note":"The popular comic-strip duo roam their many worlds in search of treasure and adventure, approaching warp speed, fighting off killer bicycles, conducting dad polls, and creating legions of snowmen and other not-so-alien beings","staffOnly":false,"instanceNoteTypeId":"fe19bae4-da28-472b-be90-d442e2428ead"},{"note":"GIFT; Bruce Lancaster; 2018; NjMD","staffOnly":false,"instanceNoteTypeId":"fe19bae4-da28-472b-be90-d442e2428ead"}], "administrativeNotes":[], "modeOfIssuanceId":"78df0873-a777-43b0-ae99-9c84faac3e4d","previouslyHeld":false,"discoverySuppress":false,"statisticalCodeIds":[],"statusUpdatedDate":"2023-03-28T11:50:21.777+0000", "tags":{"tagList":[]},"metadata":{"createdDate":"2023-03-28T11:50:21.777+00:00","createdByUserId":"308a01e4-1108-4ea3-aa25-8b1b32cf7643","updatedDate":"2023-03-28T11:50:22.083+00:00","updatedByUserId":"308a01e4-1108-4ea3-aa25-8b1b32cf7643"},"holdingsRecords2":[],"natureOfContentTermIds":[]}],

"totalRecords": 1,

"resultInfo": {"totalRecords":1,"facets":[],"diagnostics":[]}

}

EBSCO USER GROUP

caveats and disclaimers

Any and all mistakes and/or misunderstandings contained in the following are mine and mine alone.

CAUTION! Proceed at your own risk.

Measure twice, cut once.

TIMTOWTDI

“Let’s be careful out there.” ~ Sergeant Phil Esterhaus

Hill Street Blues

Table of Contents

each chapter includes:

- a brief description
- a comparative list of steps (aka clicks)
- screen shots
- “how it works”
- flow charts

post permissions

post permissions

Sometimes APIs don't work because the username does not yet have permission to do what needs to be done. The required permission can be copied and pasted from the error message into the web app.

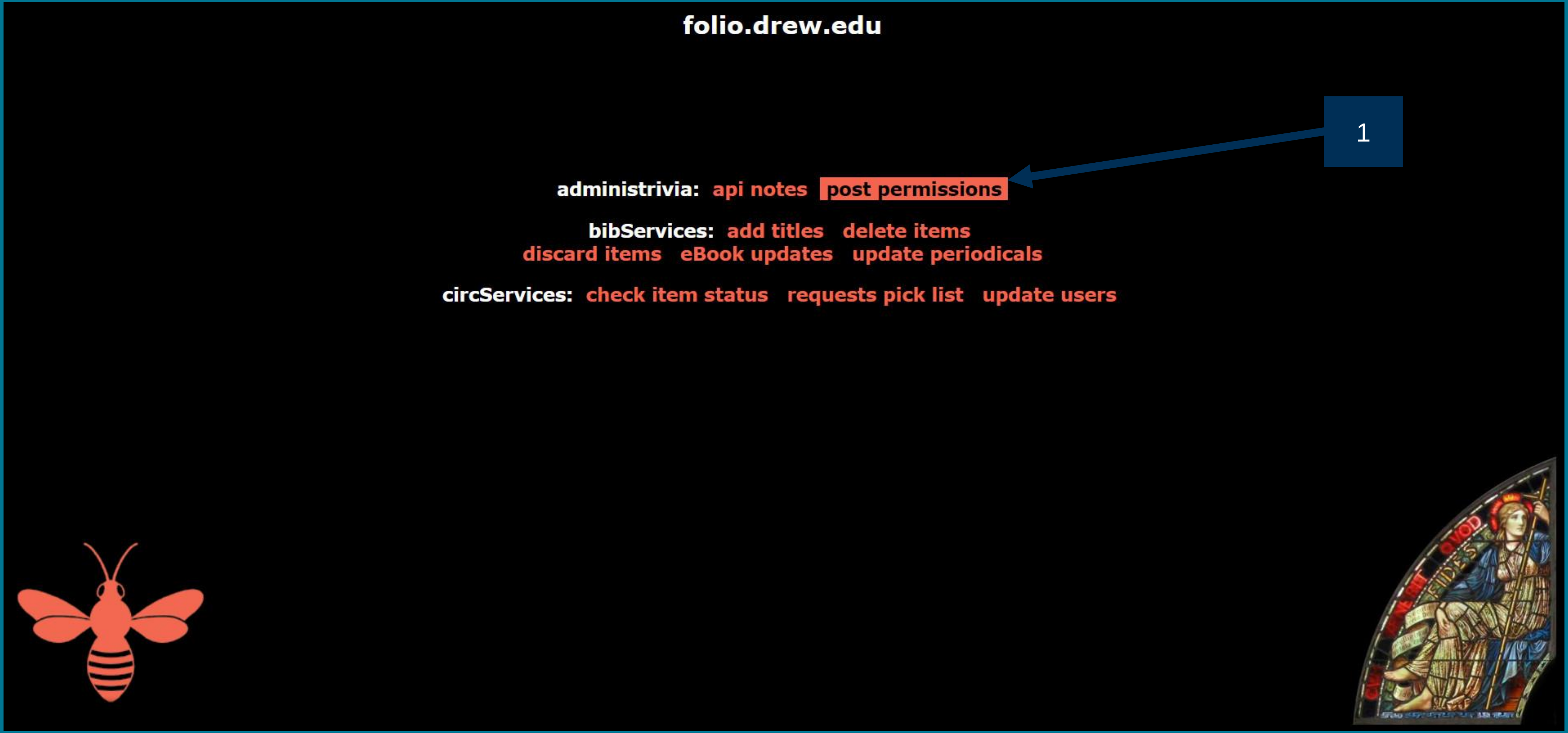
hmm: I wonder why the folio_admin permission set can't do everything?

via the UI

1. Apps
2. Users
3. search…
4. Actions
5. Edit
6. scroll down
7. open User Permissions
8. scroll down
9. Add permission
10. search…
11. check a box
12. Save & close
13. Save & close

via the web app

1. post permissions
2. copy the wanted permission from the error message
3. and paste it into the permission text box
   - the default username is admin
4. make it so



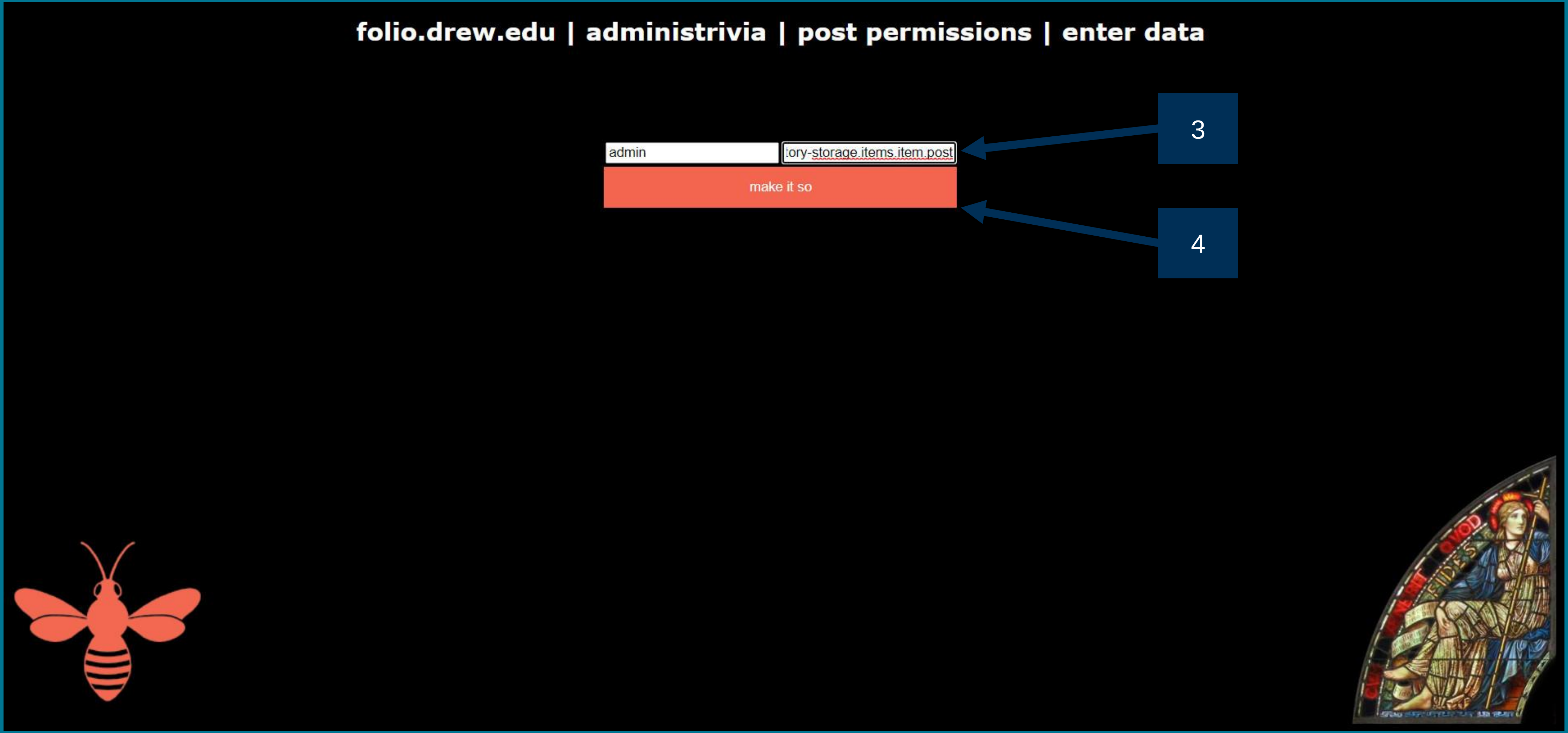
folio.drew.edu

administrivia: api notes post permissions

bibServices: add titles delete items
discard items eBook updates update periodicals

circServices: check item status requests pick list update users

EBSCO USER GROUP



folio.drew.edu | administrivia | post permissions | enter data

admin

:ory-storage.items.item.post

make it so

EBSCO USER GROUP

folio.drew.edu | administrivia | post permissions | make it so

POST https://okapi-drew.folio.ebsco.com/perms/users/3cdeb88e-b712-4e42-b192-f9c9fd9bfb8a/permissions result:

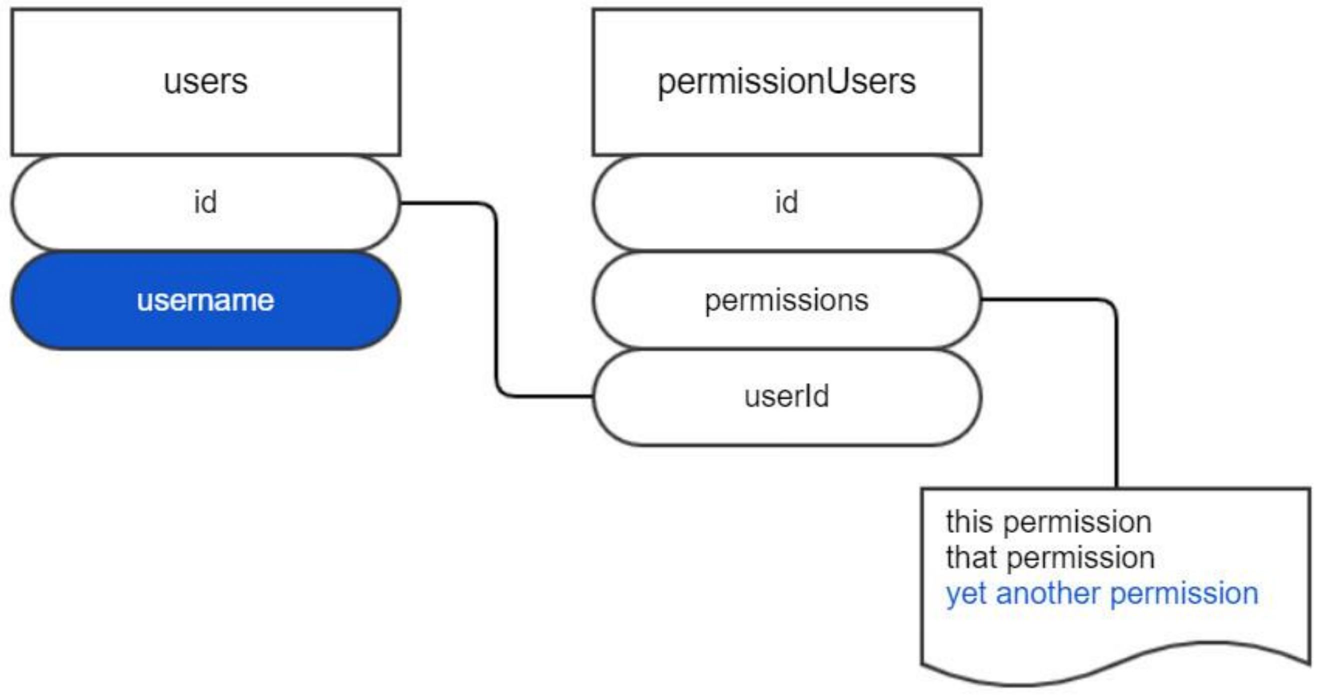
```
{
  "permissionName" : "inventory-storage.items.item.post"
}
```

how it works

- makeItSo.pl collects the $username and $permission from enterData.pl

1. **GET /users?query=username=="$username"** supplies the userId
2. **GET /perms/users?query=userId=="$userId"** supplies the permissionUsersId
3. **POST -d ‘{"permissionName":"$permission"}‘ /perms/users/$permissionUsersId/permissions** does the deed

how it works


users
id
username
permissionUsers
id
permissions
userId
this permission
that permission
yet another permission

requests pick list

requests pick list

This web app is all about trying to use less paper. The UI prints one request per page ~ we wanted as many as can fit on each page. They are sorted by location and call number

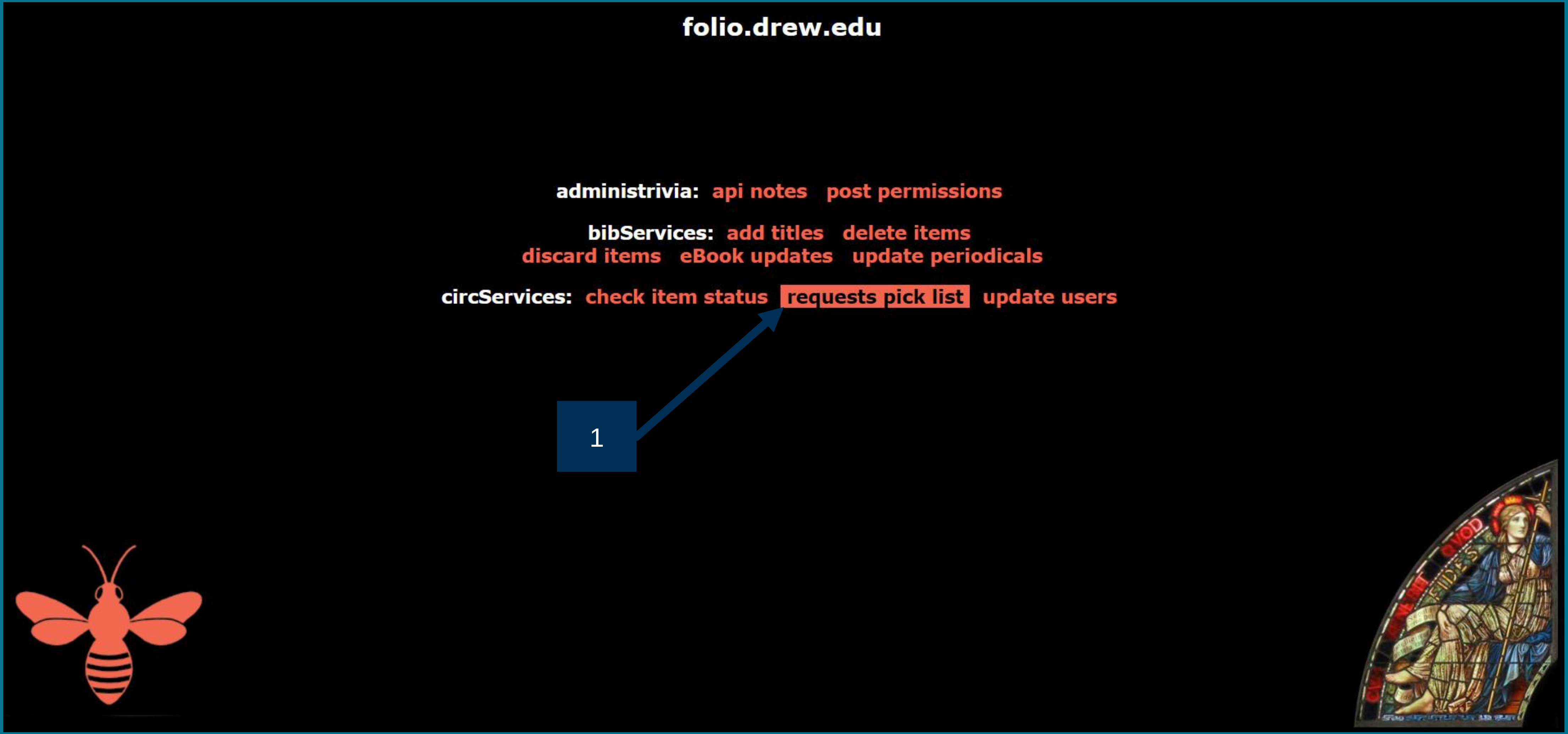
via the UI

1. Apps
2. Requests
3. Request type: Pages
4. Request status: Open – not yet filled
5. Actions
6. Print pick slips for circulation and reserves

via the web app

1. requests pick list
2. right click
3. Print…

EBSCO USER GROUP



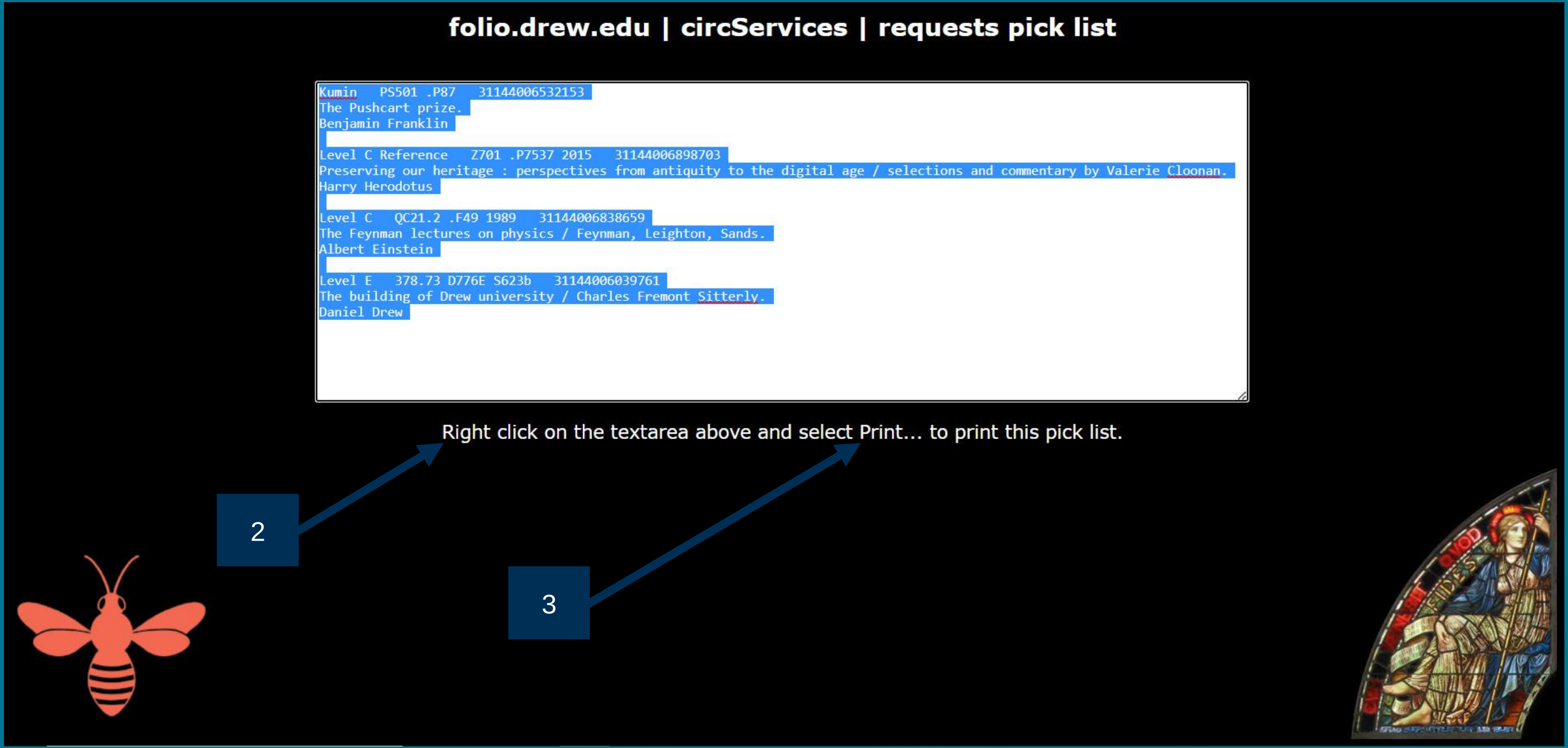
folio.drew.edu

administrivia: api notes post permissions

bibServices: add titles delete items
discard items eBook updates update periodicals

circServices: check item status requests pick list update users

EBSCO USER GROUP



folio.drew.edu | circServices | requests pick list

```
Kumin   PS501 .P87   31144006532153
The Pushcart prize.
Benjamin Franklin

Level C Reference   Z701 .P7537 2015   31144006898703
Preserving our heritage : perspectives from antiquity to the digital age / selections and commentary by Valerie Cloonan.
Harry Herodotus

Level C   QC21.2 .F49 1989   31144006838659
The Feynman lectures on physics / Feynman, Leighton, Sands.
Albert Einstein

Level E   378.73 D776E S623b   31144006039761
The building of Drew university / Charles Fremont Sitterly.
Daniel Drew
```

Right click on the textarea above and select Print... to print this pick list.

EBSCO USER GROUP

how it works

1. **GET -d 'limit=0' /request-storage/requests?query=requestType==Page**

   this returns the number of requests to be considered (aka totalRecords)

2. **GET -d 'limit=$totalRecords' /request-storage/requests?query=requestType==Page**

   this returns all of the “Page” requests so that they can be sifted to find those with an “Open - Not yet filled” status

   it also supplies the item's barcode, holdingsRecordId, requester.firstName, requester.lastName, and title

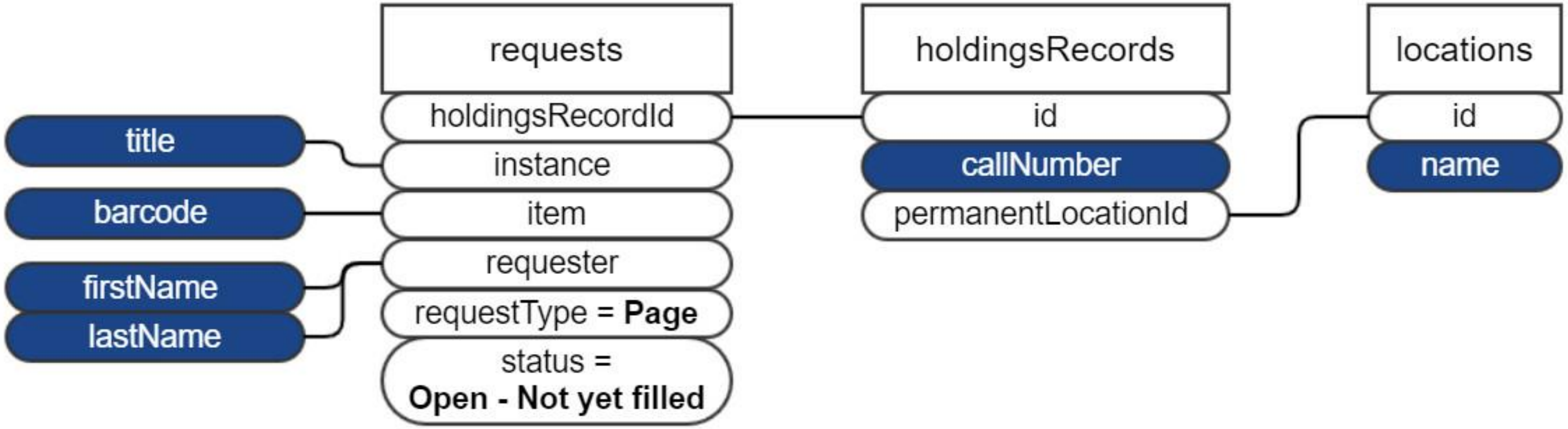
3. **GET /holdings-storage/holdings/$holdingsRecordId**

   this supplies the location and callNumber

- Perl sorts the list on location and call number and delivers it as a textarea with instructions for printing

EBSCO USER GROUP

how it works


requests
holdingsRecordId
instance
item
requester
requestType = **Page**
status = **Open - Not yet filled**
holdingsRecords
id
callNumber
permanentLocationId
locations
id
name
title
barcode
firstName
lastName

check item status

check item status

This brings a number of data points together in one place:

- author, title, publication info
- location, call number, (barcode), status
- loan date, due date,

  status, patron name and barcode

via the UI

1. Apps
2. Inventory
3. Items
4. Barcode (not Keyword)
5. enter barcode
6. Search
7. open the holdings accordion

…et voila re status

8. Circulation log
9. enter Item barcode
10. apply

…et voila re loan history

via the web app

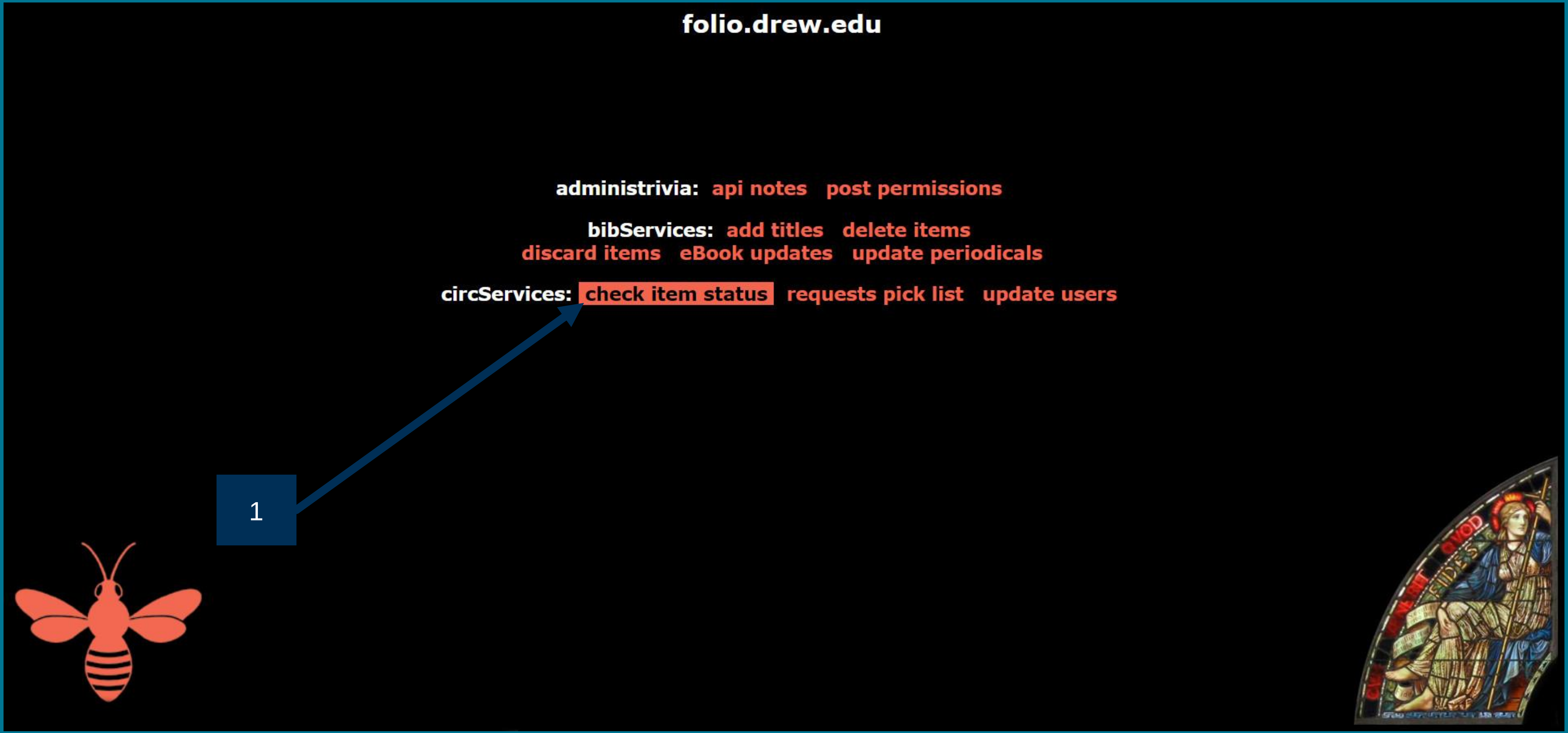
1. check item status
2. enter item barcode

…et voila re status

3. View loans

…et voila re loan history

EBSCO USER GROUP

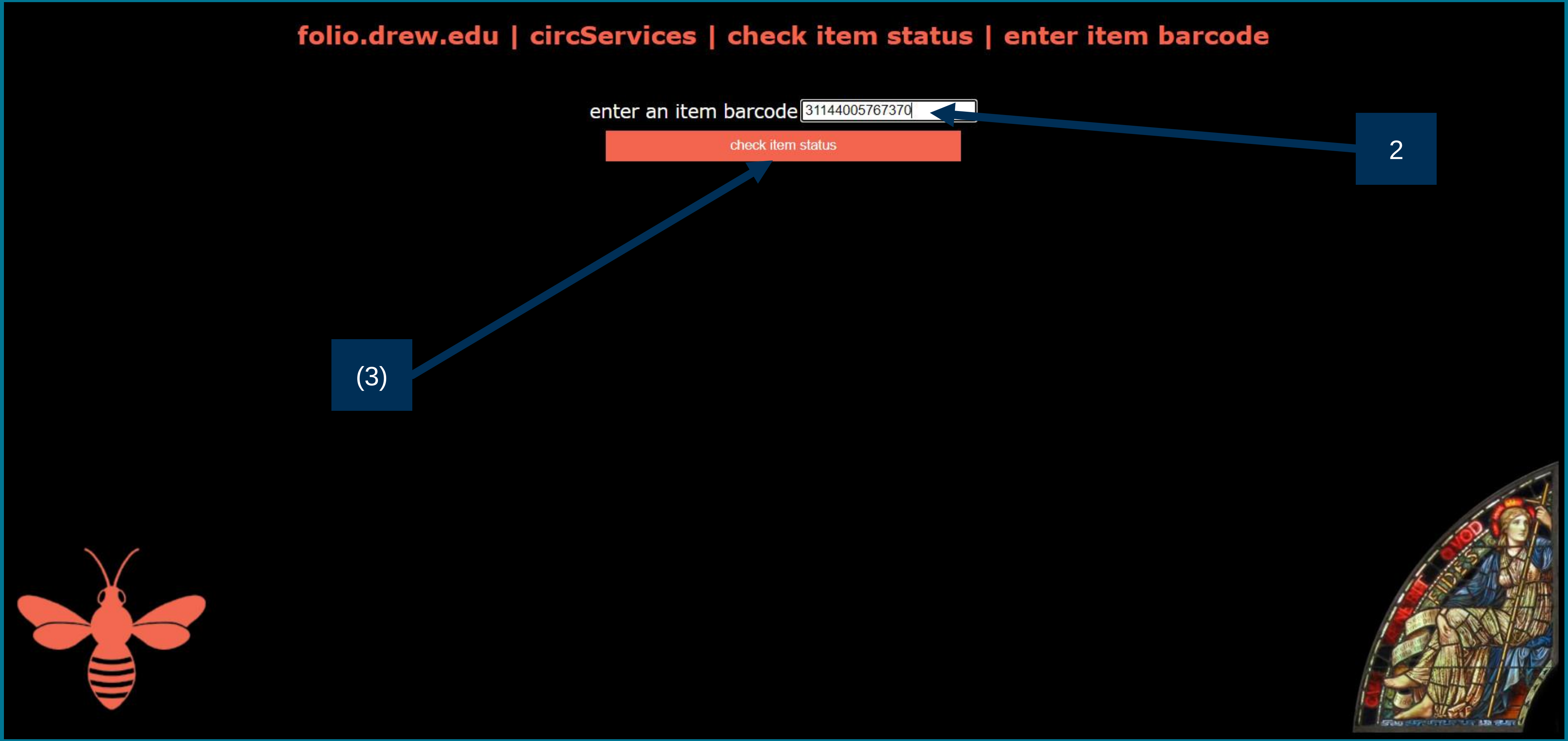


folio.drew.edu

administrivia: api notes post permissions

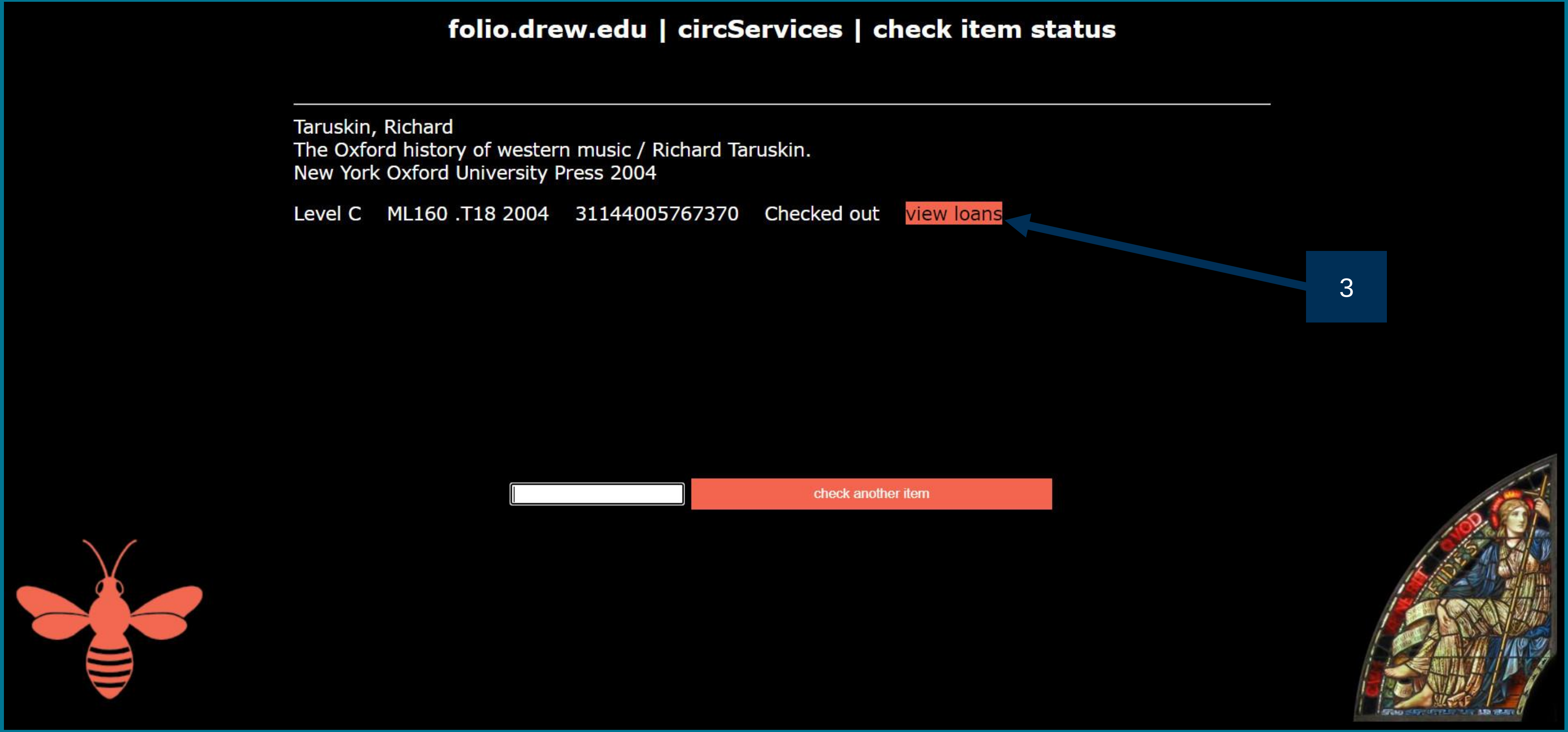
bibServices: add titles delete items
discard items eBook updates update periodicals

circServices: check item status requests pick list update users



folio.drew.edu | circServices | check item status | enter item barcode

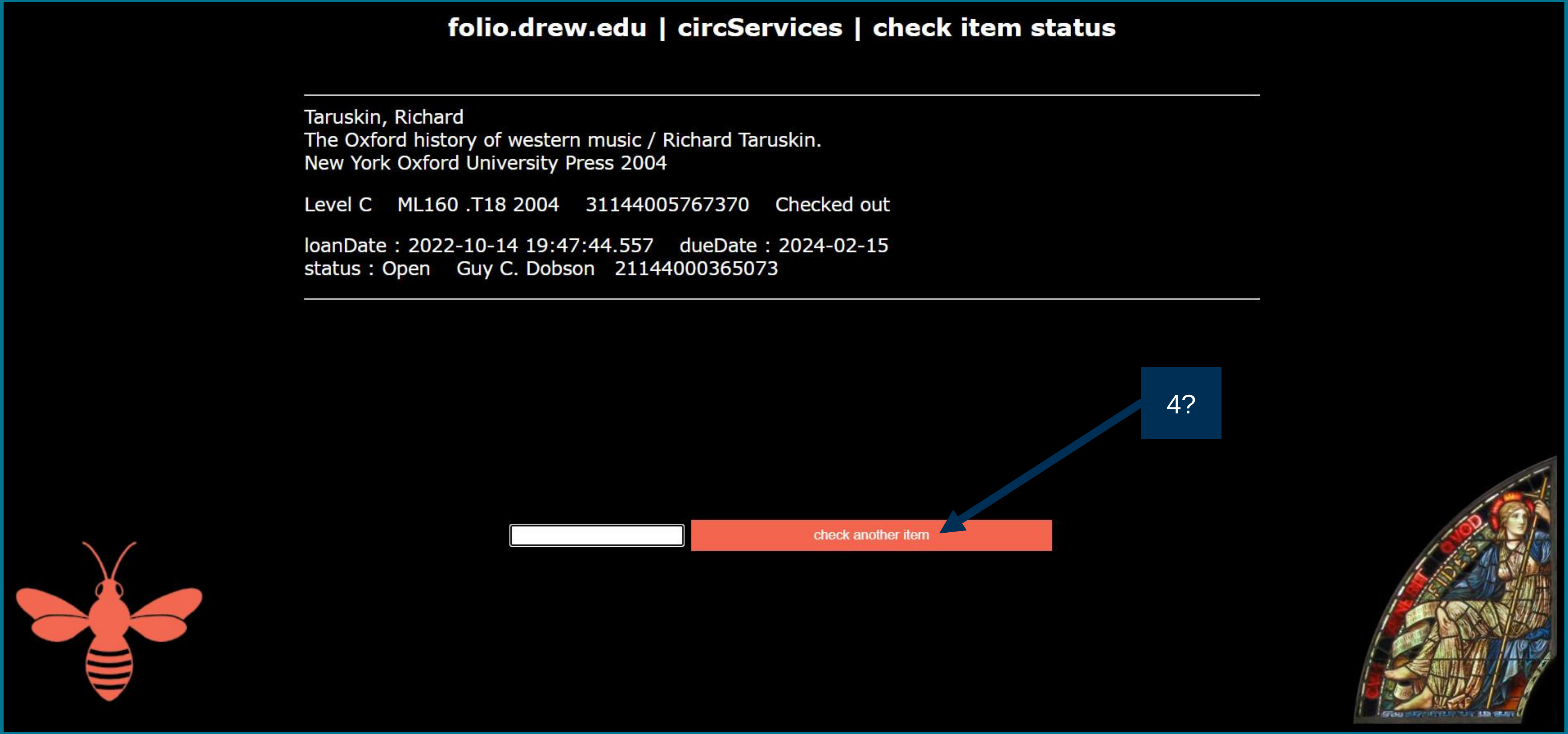
enter an item barcode 31144005767370



folio.drew.edu | circServices | check item status

Taruskin, Richard
The Oxford history of western music / Richard Taruskin.
New York Oxford University Press 2004

Level C ML160 .T18 2004 31144005767370 Checked out view loans



folio.drew.edu | circServices | check item status

Taruskin, Richard
The Oxford history of western music / Richard Taruskin.
New York Oxford University Press 2004

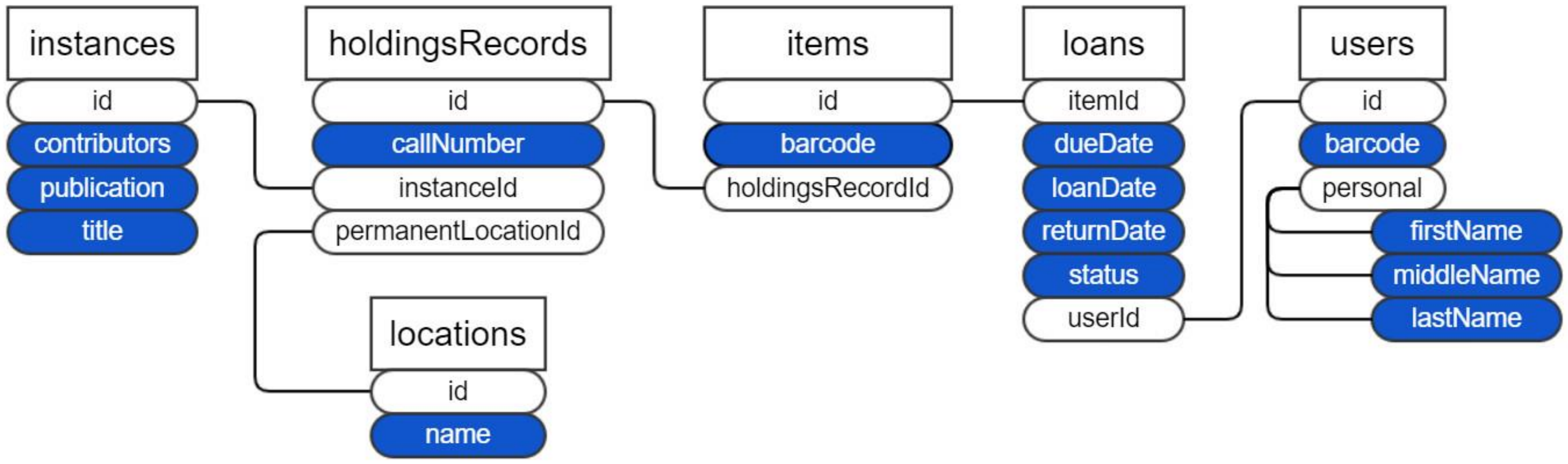
Level C   ML160 .T18 2004   31144005767370   Checked out

loanDate : 2022-10-14 19:47:44.557   dueDate : 2024-02-15
status : Open   Guy C. Dobson   21144000365073

how it works

- **GET -G -d 'limit=1000' /locations** provides location ids and names so that the permanentLocationId can be translated
- **GET /item-storage/items?query=barcode=="$barcode"** provides holdingsRecordId, itemId, and status.name
- **GET /holdings-storage/holdings/$holdingsRecordId** provides callNumber, instanceId, and permanentLocationId
- **GET /instance-storage/instances/$instanceId** provides title, contributors(s), and publication info
- **GET /circulation/loans?query=itemId=="$itemId"** provides dueDate, loanDate, returnDate, status.name, and userId
- **GET /users?query=id=="$userId"** provides patron's name and barcode
- Perl sorts the loans on loanDates
- JavaScript hides the loans from view and displays them on demand

how it works


instances
id
contributors
publication
title
holdingsRecords
id
callNumber
instanceId
permanentLocationId
items
id
barcode
holdingsRecordId
loans
itemId
dueDate
loanDate
returnDate
status
userId
users
id
barcode
personal
firstName
middleName
lastName
locations
id
name

delete items
or
update periodicals

delete items
or, update periodicals

While weeding our periodicals we found that the UI can only delete one item at a time. We were weeding a lot of items and updating our holdings statements. So many clicks! So little time!

via the UI

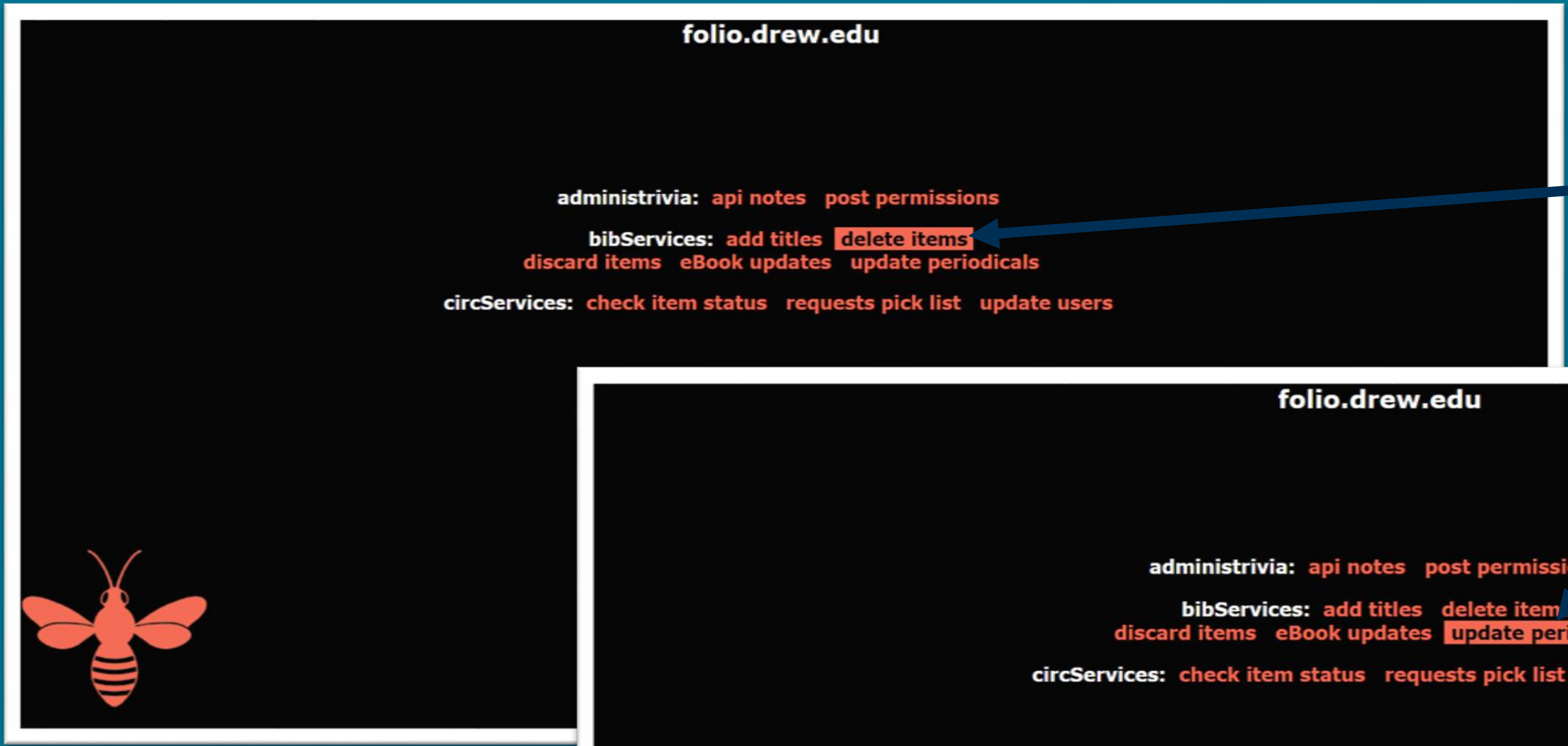
1. Apps
2. Inventory
3. Search…
4. Select
5. Open holdings accordion
6. Click on barcode
7. Actions
8. Delete
9. Delete

rinse and repeat 5-9

(total items x 5)
+ 5
= total clicks

via the web app

1. delete items
2. enter and instance hrid (or search periodicals)
3. Next…
4. select all items (or otherwise be particular)
5. delete selected items
6. are you sure?

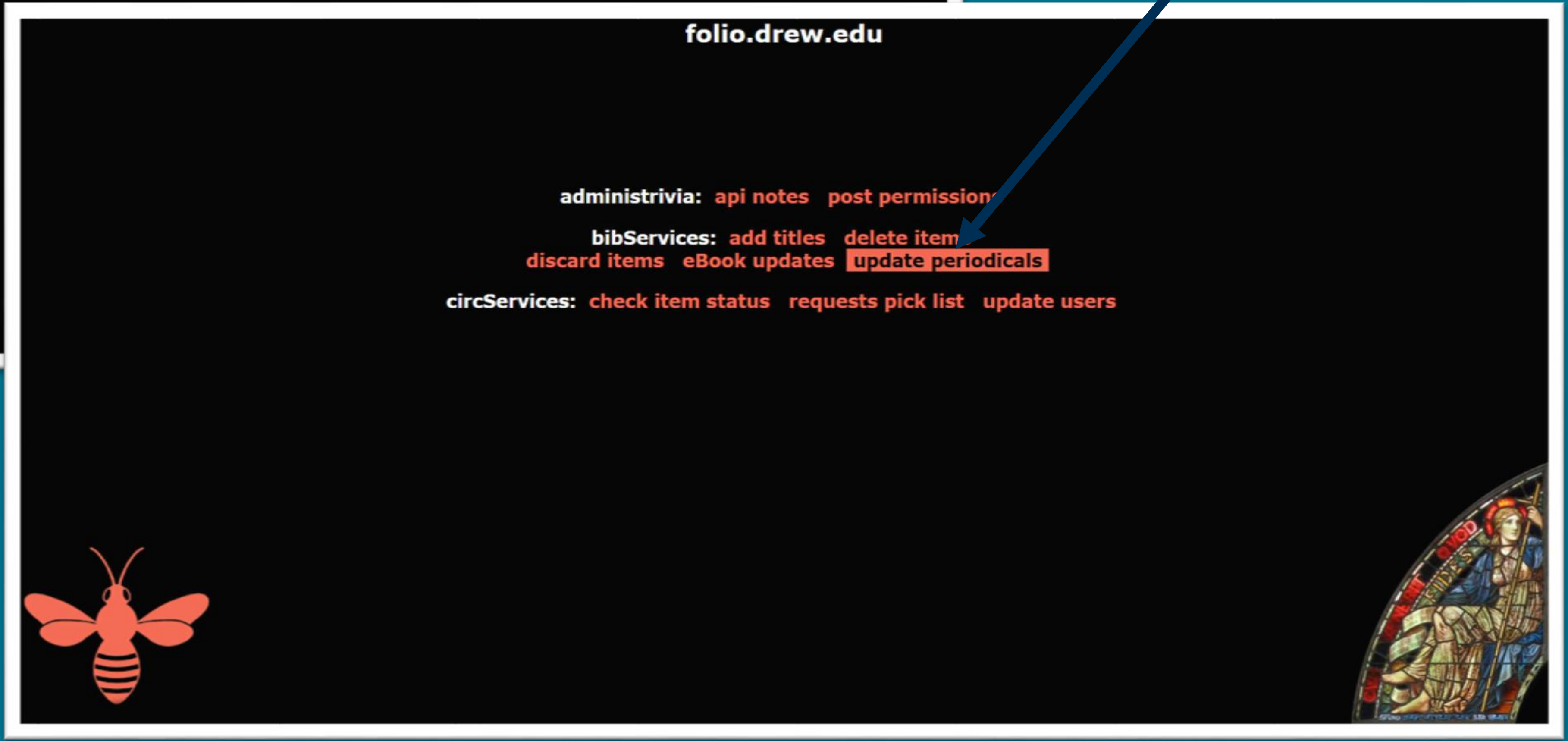


folio.drew.edu

administrivia: api notes post permissions

bibServices: add titles delete items
discard items eBook updates update periodicals

circServices: check item status requests pick list update users



folio.drew.edu

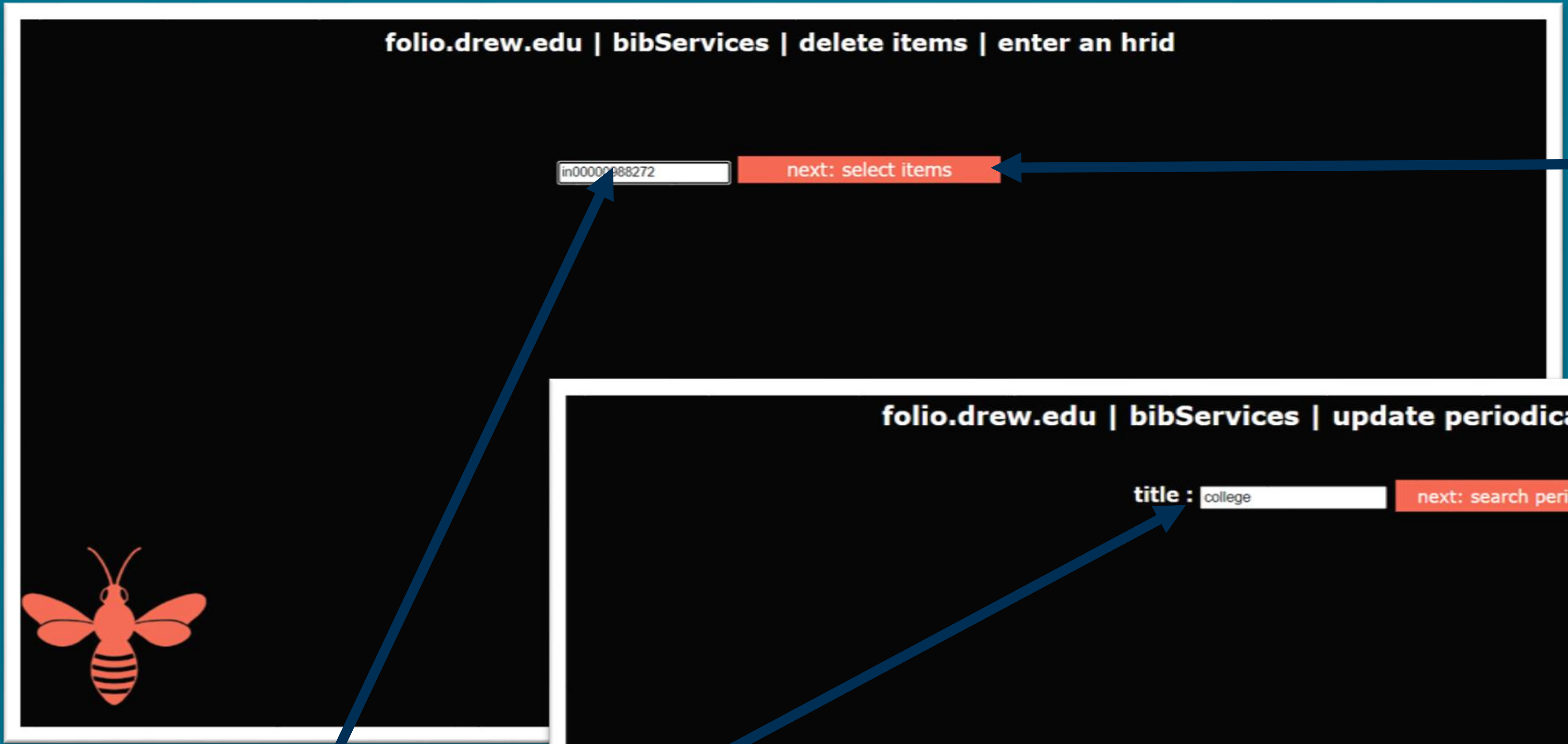
administrivia: api notes post permissions

bibServices: add titles delete items
discard items eBook updates update periodicals

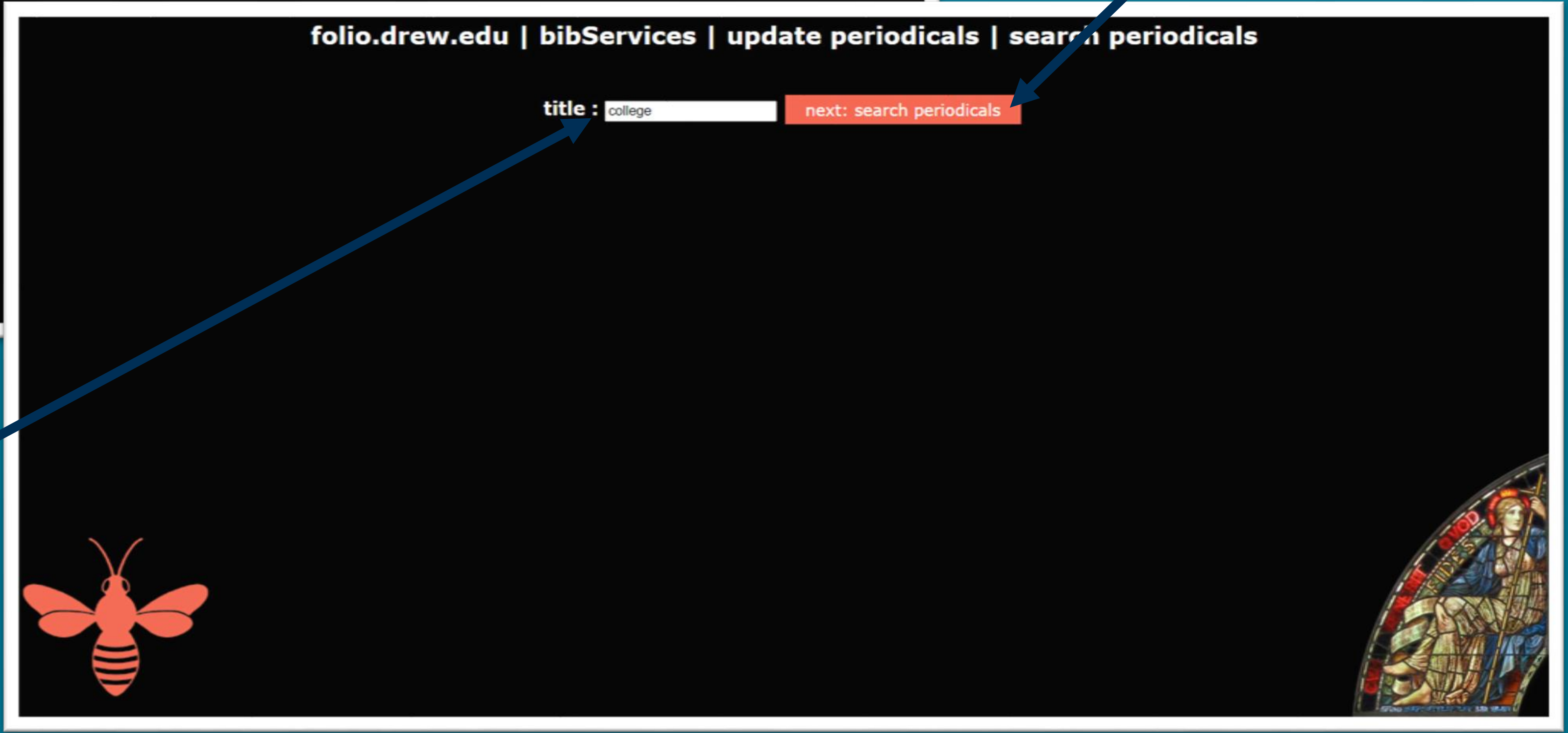
circServices: check item status requests pick list update users

1 or 1

EBSCO USER GROUP



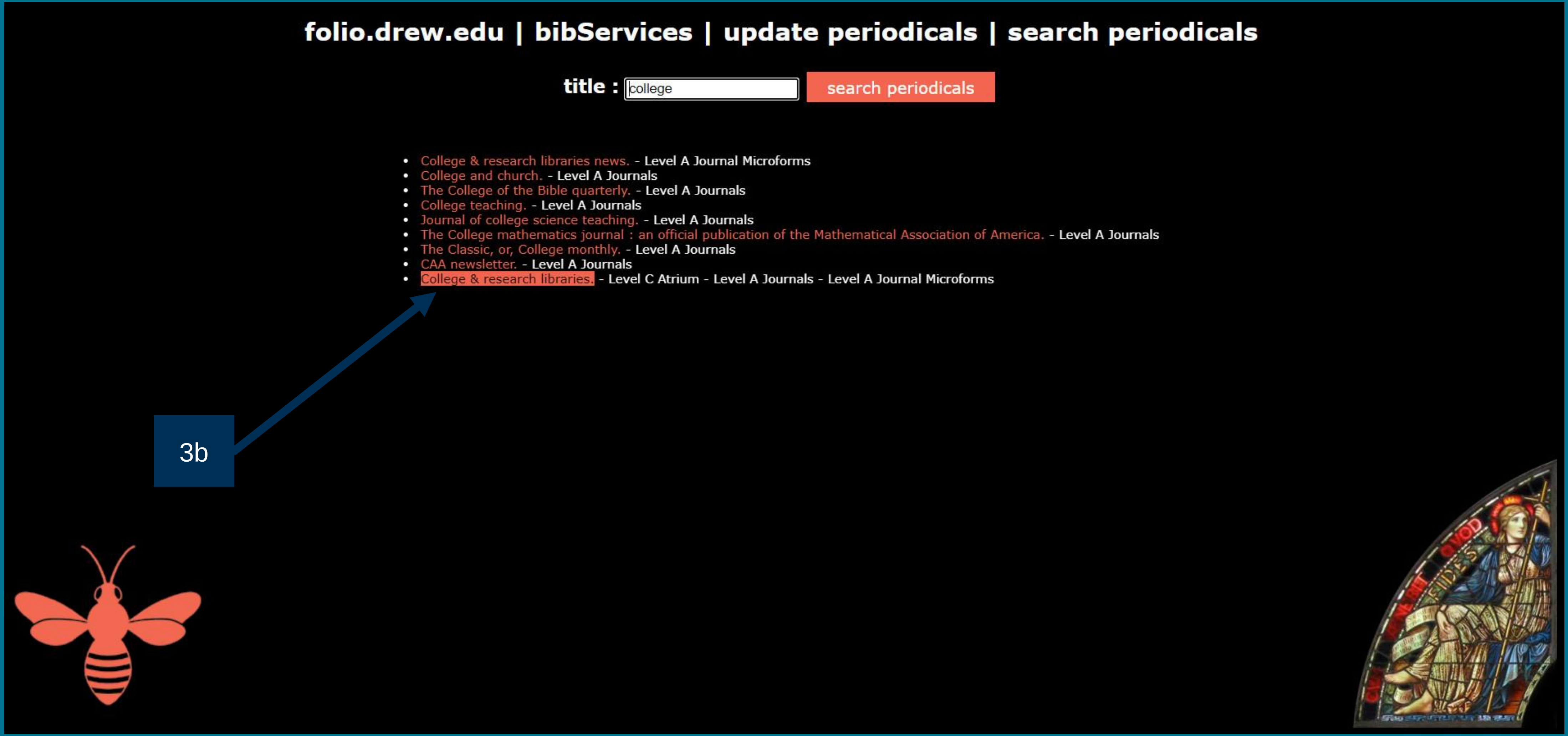
folio.drew.edu | bibServices | delete items | enter an hrid



folio.drew.edu | bibServices | update periodicals | search periodicals

3

2



folio.drew.edu | bibServices | update periodicals | search periodicals

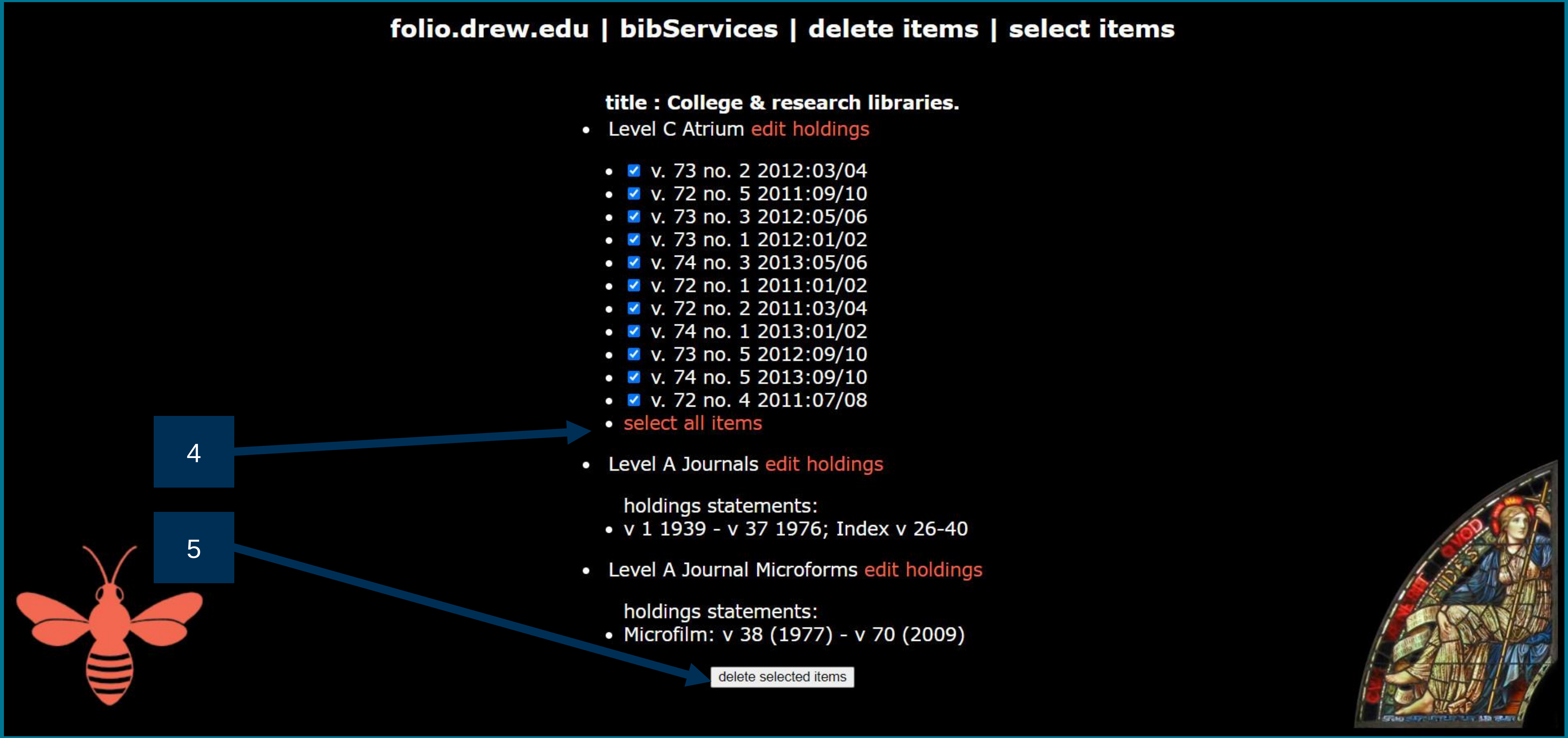
title : college

search periodicals

- College & research libraries news. - Level A Journal Microforms
- College and church. - Level A Journals
- The College of the Bible quarterly. - Level A Journals
- College teaching. - Level A Journals
- Journal of college science teaching. - Level A Journals
- The College mathematics journal : an official publication of the Mathematical Association of America. - Level A Journals
- The Classic, or, College monthly. - Level A Journals
- CAA newsletter. - Level A Journals
- College & research libraries. - Level C Atrium - Level A Journals - Level A Journal Microforms

3b

EBSCO USER GROUP


folio.drew.edu | bibServices | delete items | select items

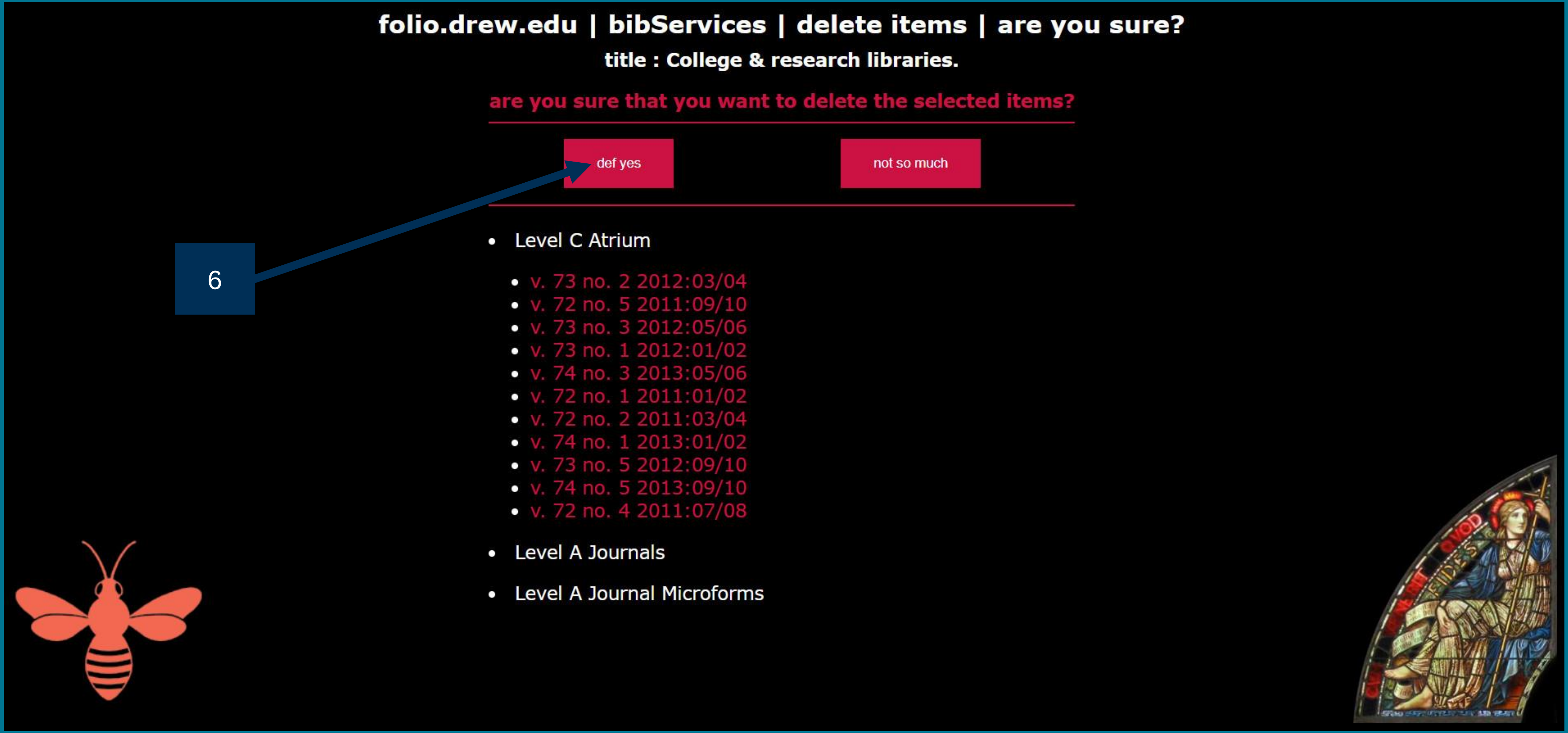
title : College & research libraries.

- Level C Atrium edit holdings
  - ✓ v. 73 no. 2 2012:03/04
  - ✓ v. 72 no. 5 2011:09/10
  - ✓ v. 73 no. 3 2012:05/06
  - ✓ v. 73 no. 1 2012:01/02
  - ✓ v. 74 no. 3 2013:05/06
  - ✓ v. 72 no. 1 2011:01/02
  - ✓ v. 72 no. 2 2011:03/04
  - ✓ v. 74 no. 1 2013:01/02
  - ✓ v. 73 no. 5 2012:09/10
  - ✓ v. 74 no. 5 2013:09/10
  - ✓ v. 72 no. 4 2011:07/08
  - select all items
- Level A Journals edit holdings

  holdings statements:
  - v 1 1939 - v 37 1976; Index v 26-40
- Level A Journal Microforms edit holdings

  holdings statements:
  - Microfilm: v 38 (1977) - v 70 (2009)

EBSCO USER GROUP



folio.drew.edu | bibServices | delete items | are you sure?

title : College & research libraries.

are you sure that you want to delete the selected items?

- Level C Atrium
  - v. 73 no. 2 2012:03/04
  - v. 72 no. 5 2011:09/10
  - v. 73 no. 3 2012:05/06
  - v. 73 no. 1 2012:01/02
  - v. 74 no. 3 2013:05/06
  - v. 72 no. 1 2011:01/02
  - v. 72 no. 2 2011:03/04
  - v. 74 no. 1 2013:01/02
  - v. 73 no. 5 2012:09/10
  - v. 74 no. 5 2013:09/10
  - v. 72 no. 4 2011:07/08
- Level A Journals
- Level A Journal Microforms

EBSCO USER GROUP

how it works

Re searchPeriodiclas.pl

- **GET /search/instances/ids?query=title="$query"** just returns instanceIds
- **GET /instance-storage/instances?query=id=="$instanceId"** provides titles HRIDs
- **GET /holdings-storage/holdings?query=instanceId=="$instanceId"** provides permanentLocationIds (we only want periodical locations)

Re selectItems.pl's:

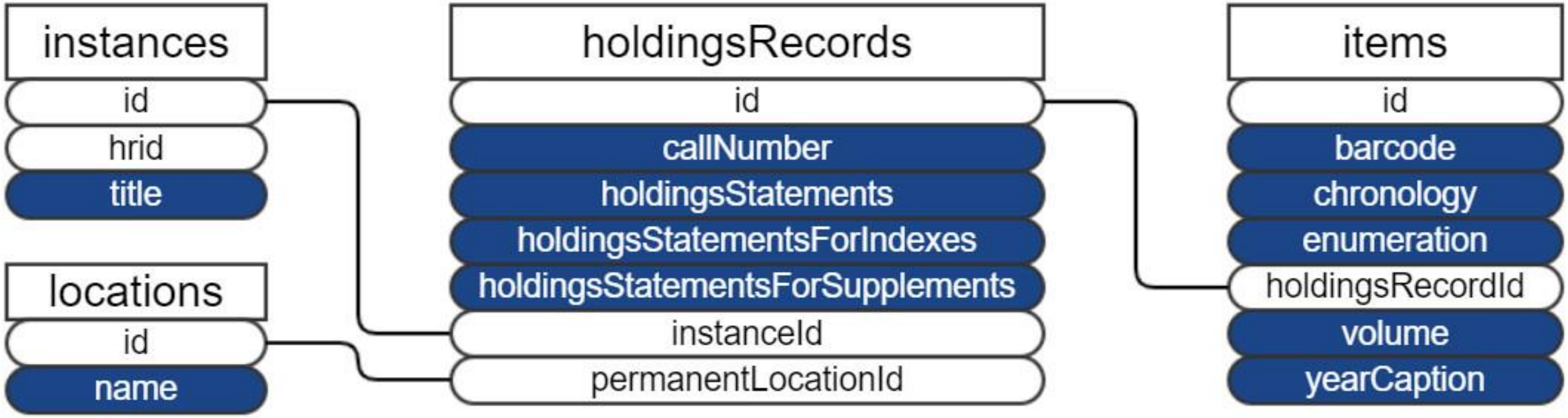
- **GET /instance-storage/instances?query=hrid==$hrid** which provides the instanceId and the title
- **GET /holdings-storage/holdings?query=instanceId==$instanceId** which provides the callNumber, holdingsRecordId, holdingsStatements, and permanentLocationId
- **GET /item-storage/items?query=holdingsRecordId==$holdingsRecordId** which provides the itemId, barcode, chronology, enumeration, volume, etc.

Links to the UI provide an opportunity to edit holdings statements and javaScript enables all items to be selected at once. The hrid along with the UUIDs of selected items are then sent to areYouSure.pl which, if you are sure, forwards the itemIds to deleteItemsAPI.pl which then, for each UUID…

- **DELETE /item-storage/items/$itemId**

EBSCO USER GROUP

how it works


instances
id
hrid
title
locations
id
name
holdingsRecords
id
callNumber
holdingsStatements
holdingsStatementsForIndexes
holdingsStatementsForSupplements
instanceId
permanentLocationId
items
id
barcode
chronology
enumeration
holdingsRecordId
volume
yearCaption

eBook updates

eBook updates

We want our eBook (& eVideo) subscriptions, DDA packages, etc to be discoverable in our Locate OPAC so this web app creates instances, holdings, and items that can be removed if/when DDA titles get dropped or a subscription is cancelled. New titles are added to a queue to be loaded from the command line. Old titles can be removed directly via the web app.

via the UI

n/a

via the web app

1. eBook updates
2. select a vendor

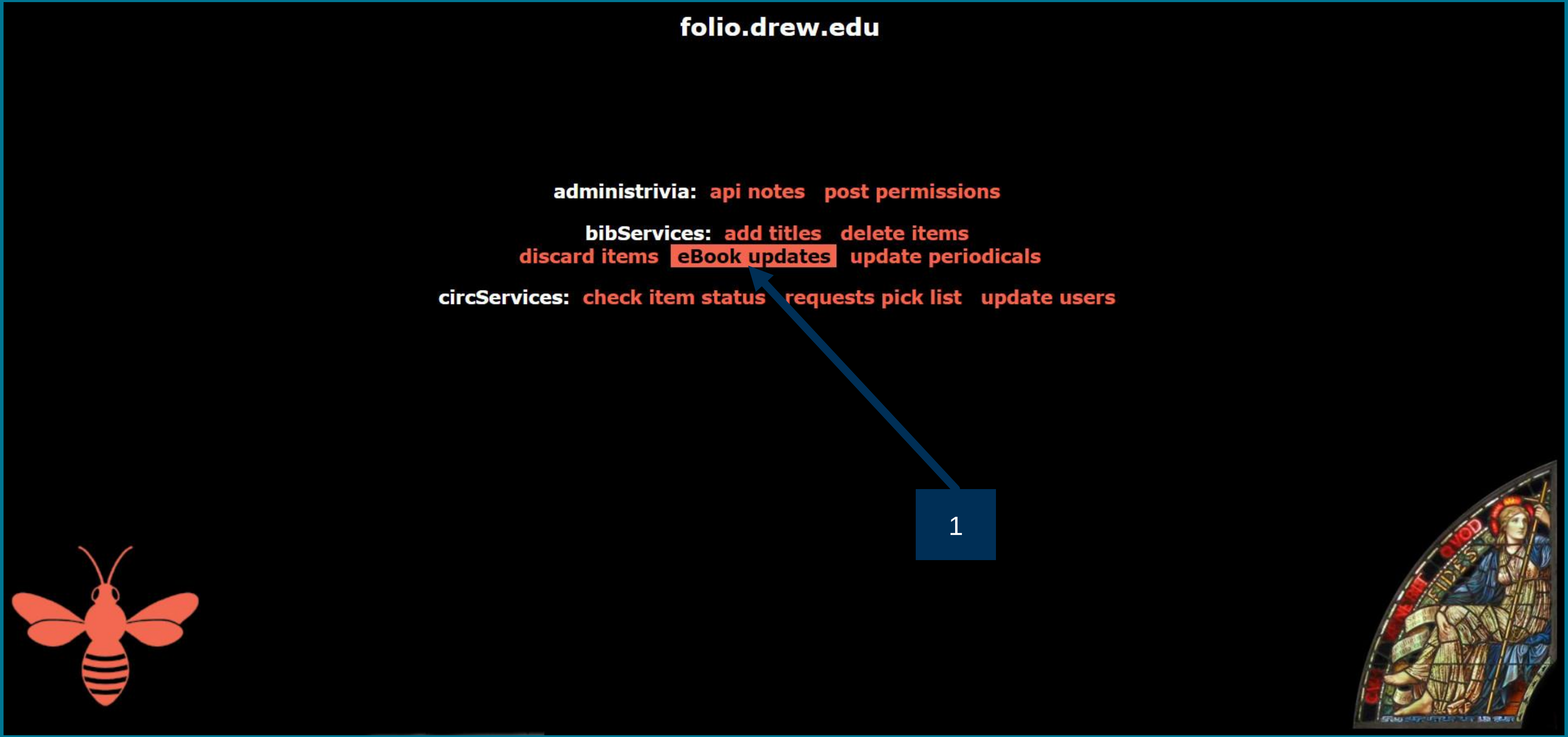
for new titles:

3. choose a file
4. make it so

For old titles:

3. remove
4. make it so
5. enter vendorId(s)
6. remove item(s), holdingsRecord(s), and instance(s)

EBSCO USER GROUP

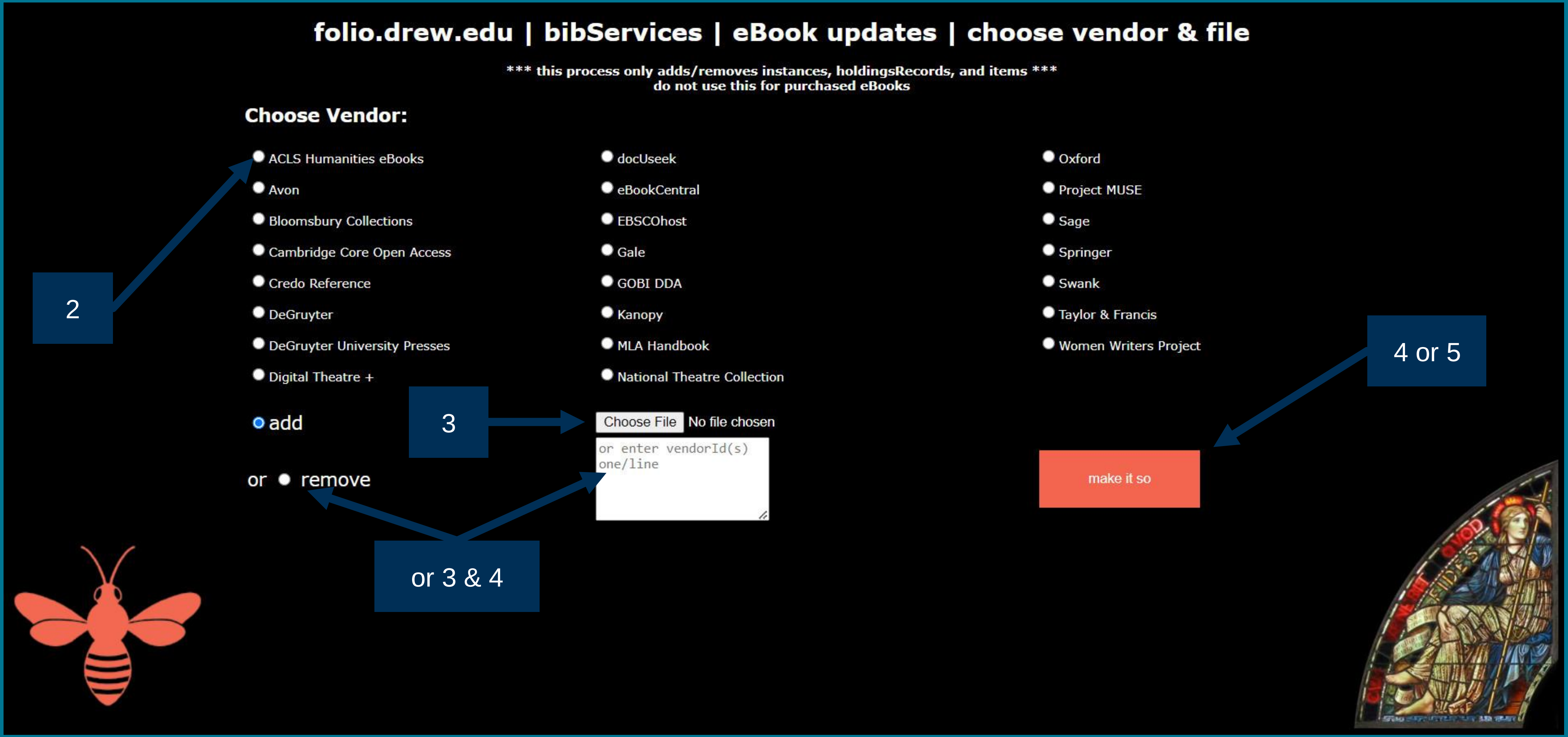


folio.drew.edu

administrivia: api notes post permissions

bibServices: add titles delete items
discard items eBook updates update periodicals

circServices: check item status requests pick list update users



folio.drew.edu | bibServices | eBook updates | choose vendor & file

*** this process only adds/removes instances, holdingsRecords, and items ***
do not use this for purchased eBooks

Choose Vendor:

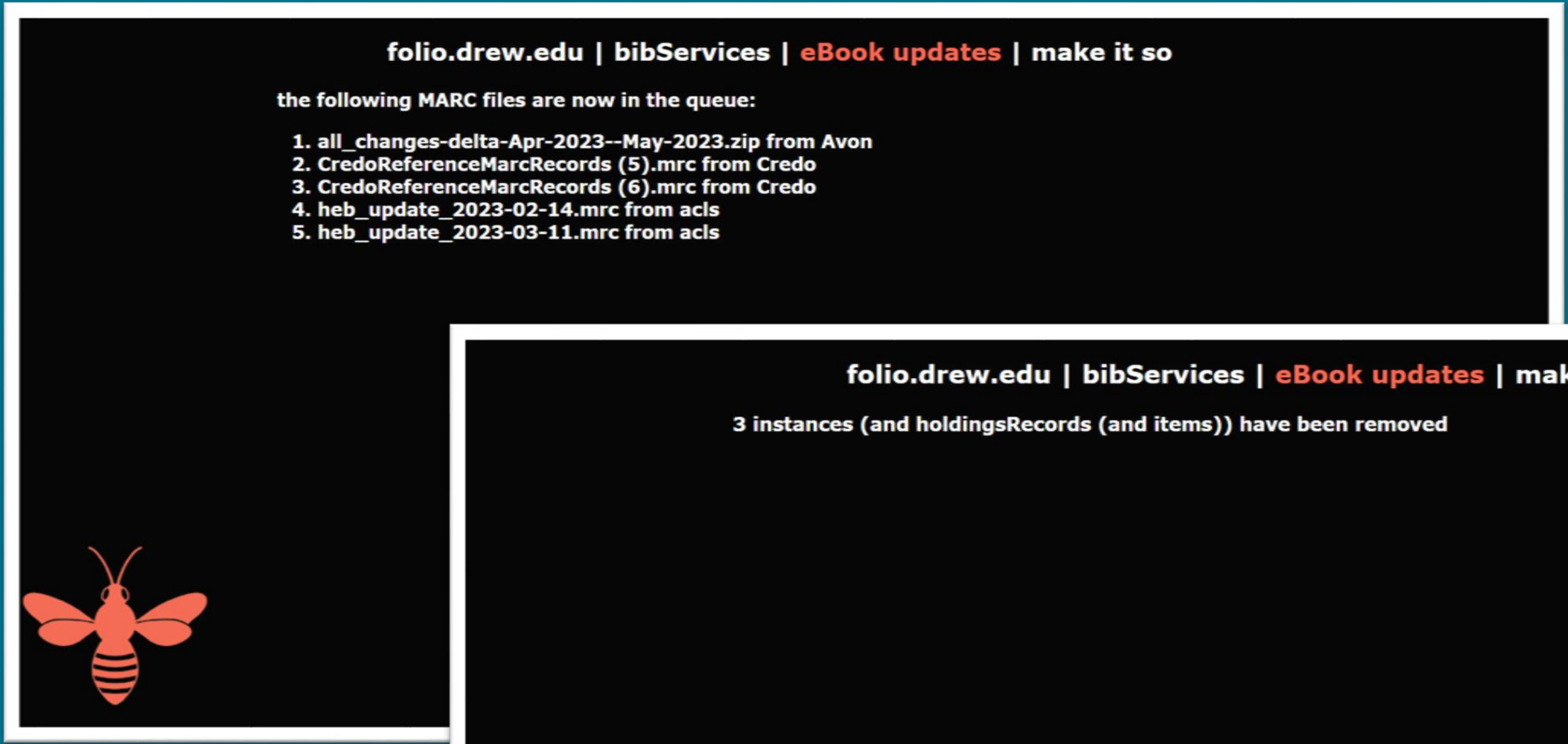
- ACLS Humanities eBooks
- Avon
- Bloomsbury Collections
- Cambridge Core Open Access
- Credo Reference
- DeGruyter
- DeGruyter University Presses
- Digital Theatre +
- docUseek
- eBookCentral
- EBSCOhost
- Gale
- GOBI DDA
- Kanopy
- MLA Handbook
- National Theatre Collection
- Oxford
- Project MUSE
- Sage
- Springer
- Swank
- Taylor & Francis
- Women Writers Project

[✓ add] or remove

Choose File No file chosen

or enter vendorId(s) one/line

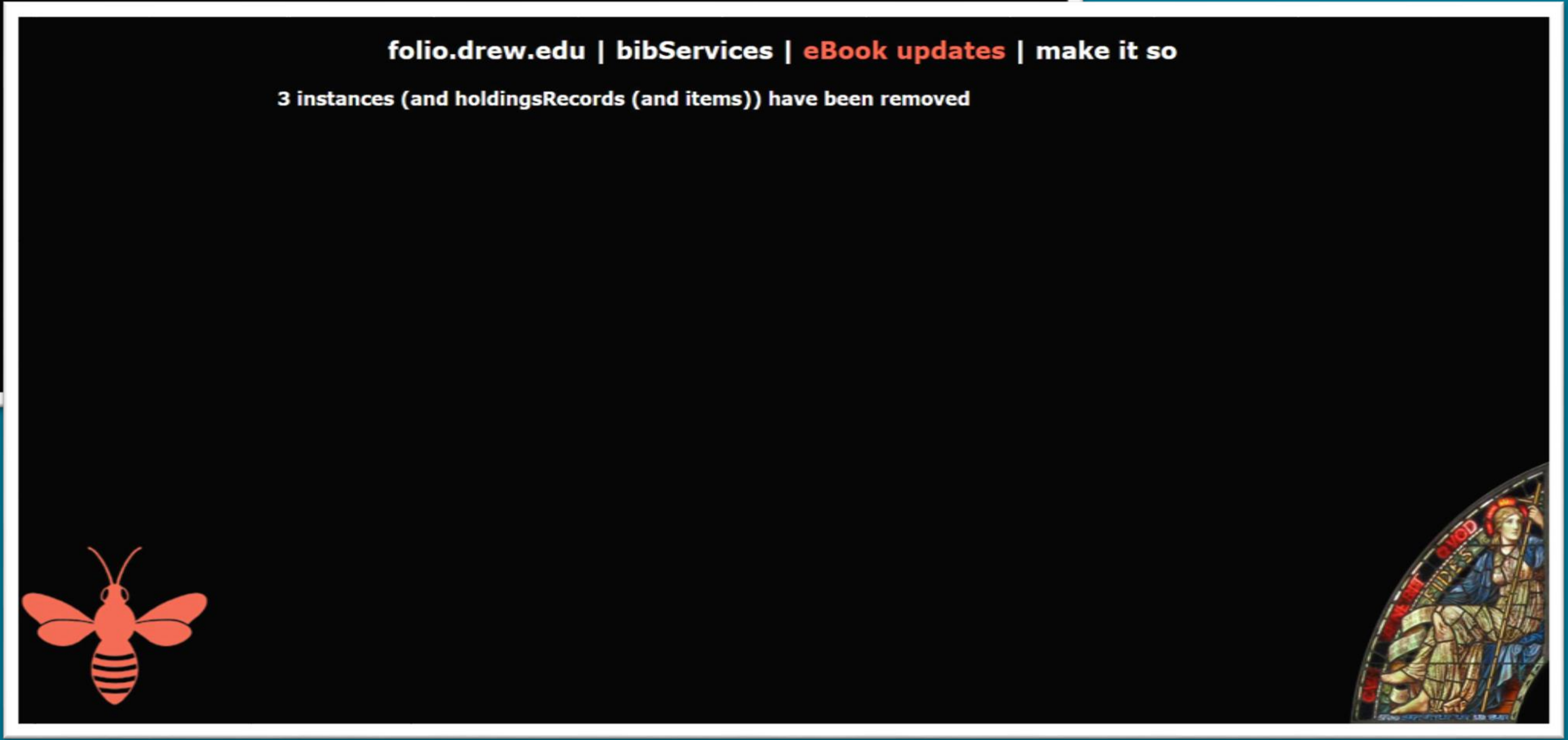
make it so



folio.drew.edu | bibServices | eBook updates | make it so

the following MARC files are now in the queue:

1. all_changes-delta-Apr-2023--May-2023.zip from Avon
2. CredoReferenceMarcRecords (5).mrc from Credo
3. CredoReferenceMarcRecords (6).mrc from Credo
4. heb_update_2023-02-14.mrc from acls
5. heb_update_2023-03-11.mrc from acls



folio.drew.edu | bibServices | eBook updates | make it so

3 instances (and holdingsRecords (and items)) have been removed

EBSCO USER GROUP

how it works

if, after selecting a vendor, a file is chosen for “add” :

- The vendor code is prepended to the filename which is then added to a queue for overnight processing which will
    - POST JSONs to /instance-storage/instances (instance.source = vendorCode), /holdings-storage/holdings, & item-storage/items
    - store the relevant vendorId, instanceId, holdingsRecordId, & itemId in a MySQL table named eBookUpdates.folioUUIDs

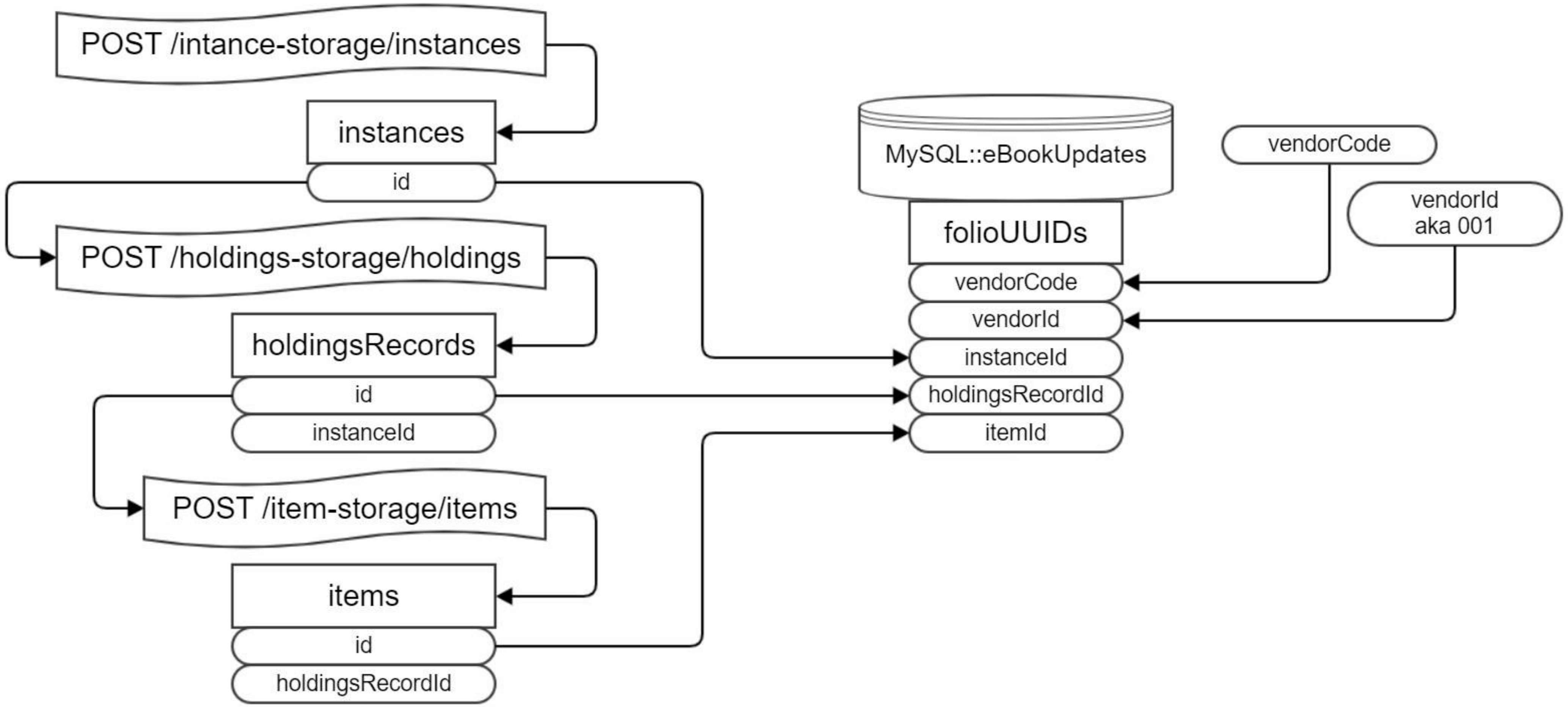
if “remove” is selected and vendorIds are listed in the textarea :

- *SELECT itemId, holdingsRecordId, instanceId FROM eBookUpdates WHERE vendorCode=$vendorCode and vendorId=$vendorId*
- **DELETE /item-storage/items/$itemId**
- **DELETE /holdings-storage/holdings/$holdingsRecordId**
- **DELETE /instance-storage/instances/$instanceId**
- *DELETE FROM eBookUpdates WHERE vendorId=$vendorId*

if “remove” is selected and an asterisk is entered in the textarea (aka remove all instances for that vendor) :

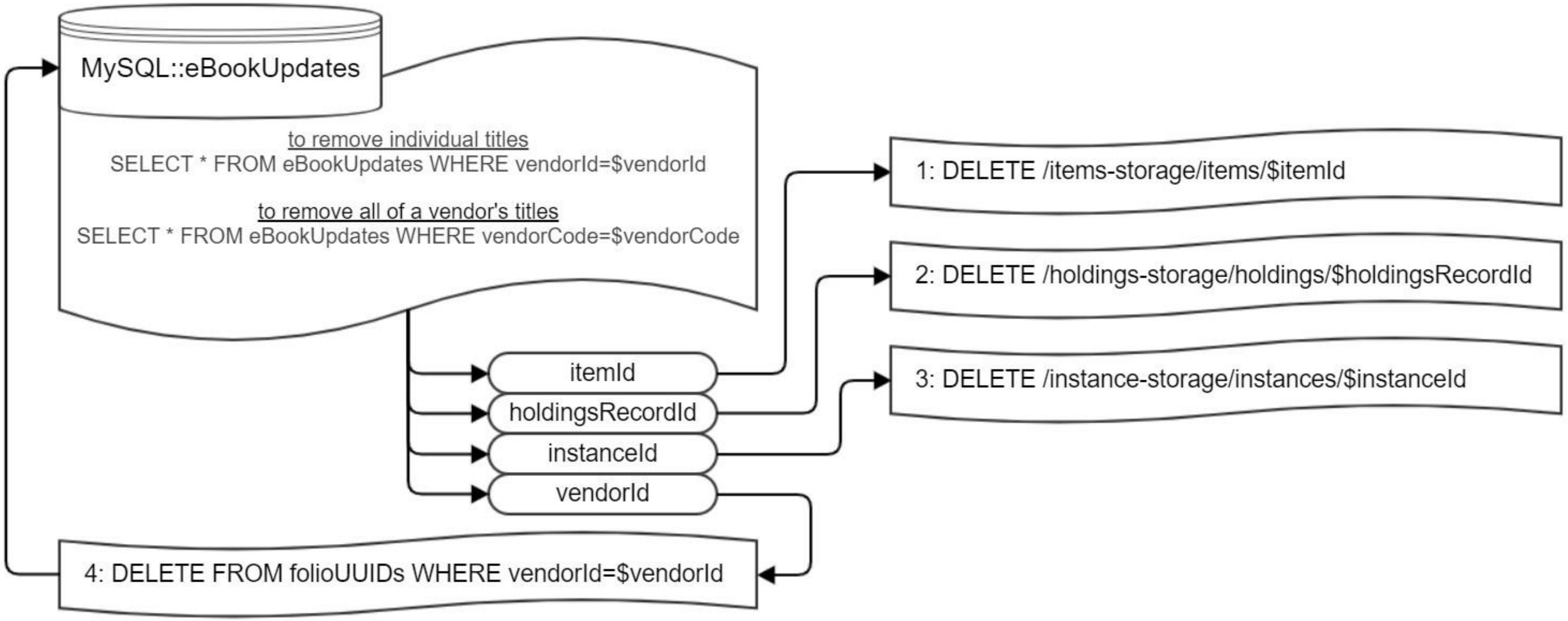
- *SELECT itemId, holdingsRecordId, instanceId FROM eBookUpdates WHERE vendorCode=$vendorCode*
- etc, etc, etc…

how it works : adding eBooks



EBSCO USER GROUP

how it works : removing eBooks


MySQL::eBookUpdates
to remove individual titles
SELECT * FROM eBookUpdates WHERE vendorId=$vendorId
to remove all of a vendor's titles
SELECT * FROM eBookUpdates WHERE vendorCode=$vendorCode
itemId
holdingsRecordId
instanceId
vendorId
1: DELETE /items-storage/items/$itemId
2: DELETE /holdings-storage/holdings/$holdingsRecordId
3: DELETE /instance-storage/instances/$instanceId
4: DELETE FROM folioUUIDs WHERE vendorId=$vendorId

EBSCO USER GROUP

update users

update users

Every once in a while we change fixed due dates and expiration dates for employee (faculty and staff), or student patronGroups. It isn't a big deal to change due dates in the UI but changing user expiration dates en masse is not an option, so, might as well do both and make have the "make it so" form guess what those patronGroups and relevant dates should be.

via the UI

re due dates

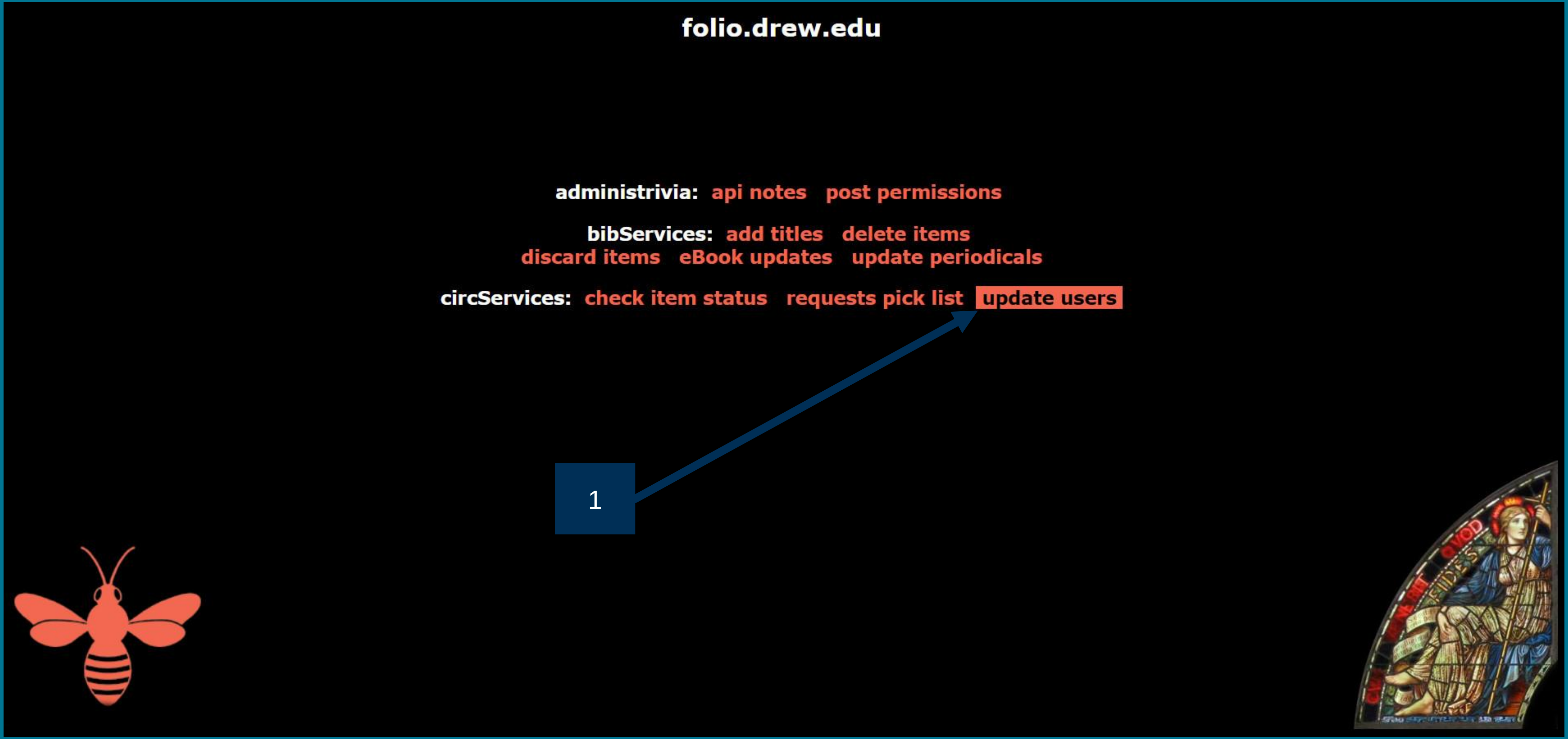
1. Apps
2. Settings
3. Circulation
4. Loans:Fixed due date schedules
5. Faculty due date
6. Edit
7. change Date from
8. change Date to
9. change Due date
10. Save & Close

re user expiration dates

n/a

via the web app

1. update users
   - change dates, maybe
2. make it so



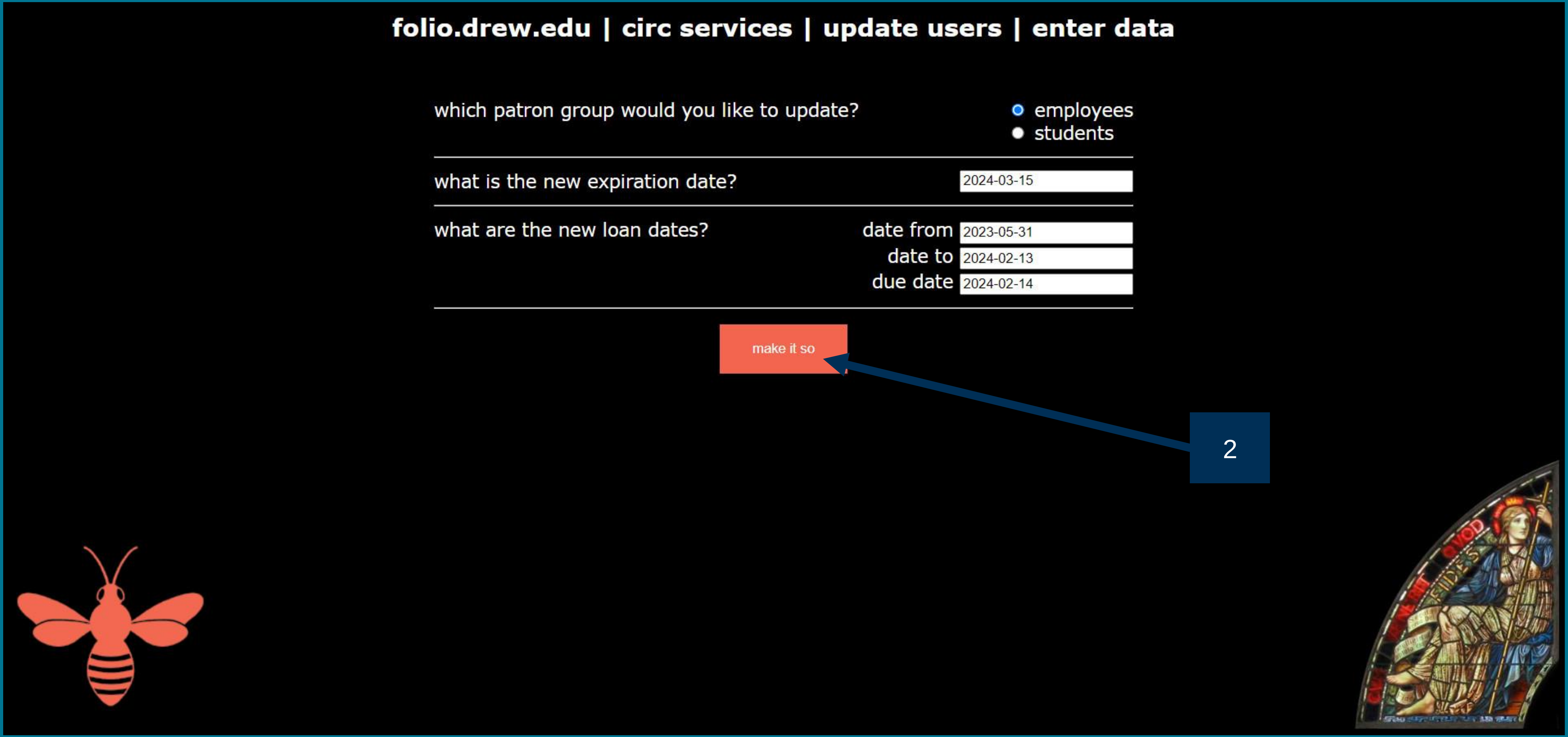
folio.drew.edu

administrivia: api notes post permissions

bibServices: add titles delete items
discard items eBook updates update periodicals

circServices: check item status requests pick list update users

EBSCO USER GROUP



folio.drew.edu | circ services | update users | enter data

which patron group would you like to update?	[✓ employees] students
what is the new expiration date?	2024-03-15
what are the new loan dates?	date from 2023-05-31 date to 2024-02-13 due date 2024-02-14

how it works

The enterData.pl form knows what day it is and makes an educated guess re which patron group you want to fuss with and what the new due and expiration dates should be. You can, of course, change anything that it doesn't get right.

Re expiration dates:

- If a person currently has either an employee or a student "role" in Banner (and matches the patronGroup in question):

1. **GET /users?query=barcode==$barcode** returns a JSON document which, after massaging with regex to replace…
    - **"active":false** with **"active":true**, and
    - the old expiration date (whatever it is) with the new expiration date

   …can be used to replace the user record via:

2. **PUT -d '$json' /users/$userId**

EBSCO USER GROUP

how it works

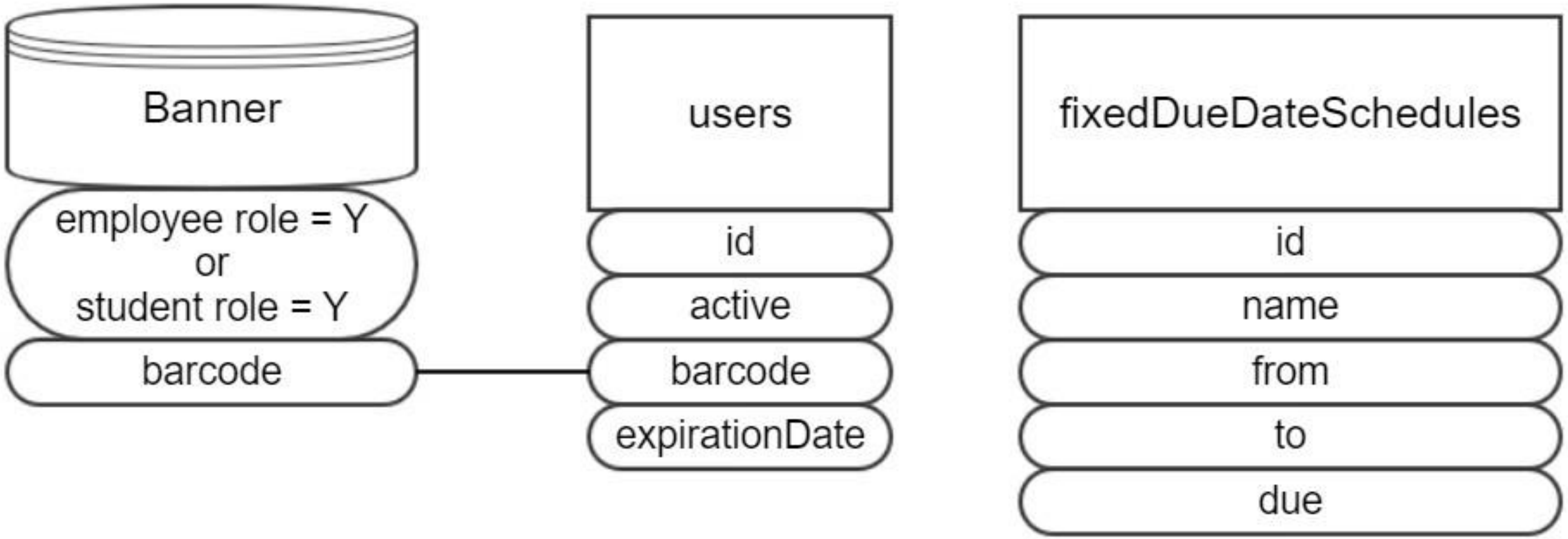
Re due dates (aka fixed due date schedules) :

3. **GET /fixed-due-date-schedule-storage/fixed-due-date-schedules?query=name="faculty*"** returns a JSON document which, after massaging with regex to replace…
    - **"from" : "2023-02-03T00:00:00.000+00:00",** with **"from" : "2024-02-03T00:00:00.000+00:00",**
    - **"to" : "2024-02-14T04:59:59.000+00:00",** with **"to" : "2025-02-14T04:59:59.000+00:00",**
    - and **"due" : "2024-02-15T04:59:59.000+00:00"** with **"due" : "2025-02-15T04:59:59.000+00:00"**

   …can be used to replace the fixedDueDateSchedules record via:

4. **PUT -d '$json' /fixed-due-date-schedule-storage/fixed-due-date-schedules/$fixedDueDateScheduleId**

how it works



EBSCO USER GROUP

add titles
(along with a call number and a barcode)

add titles

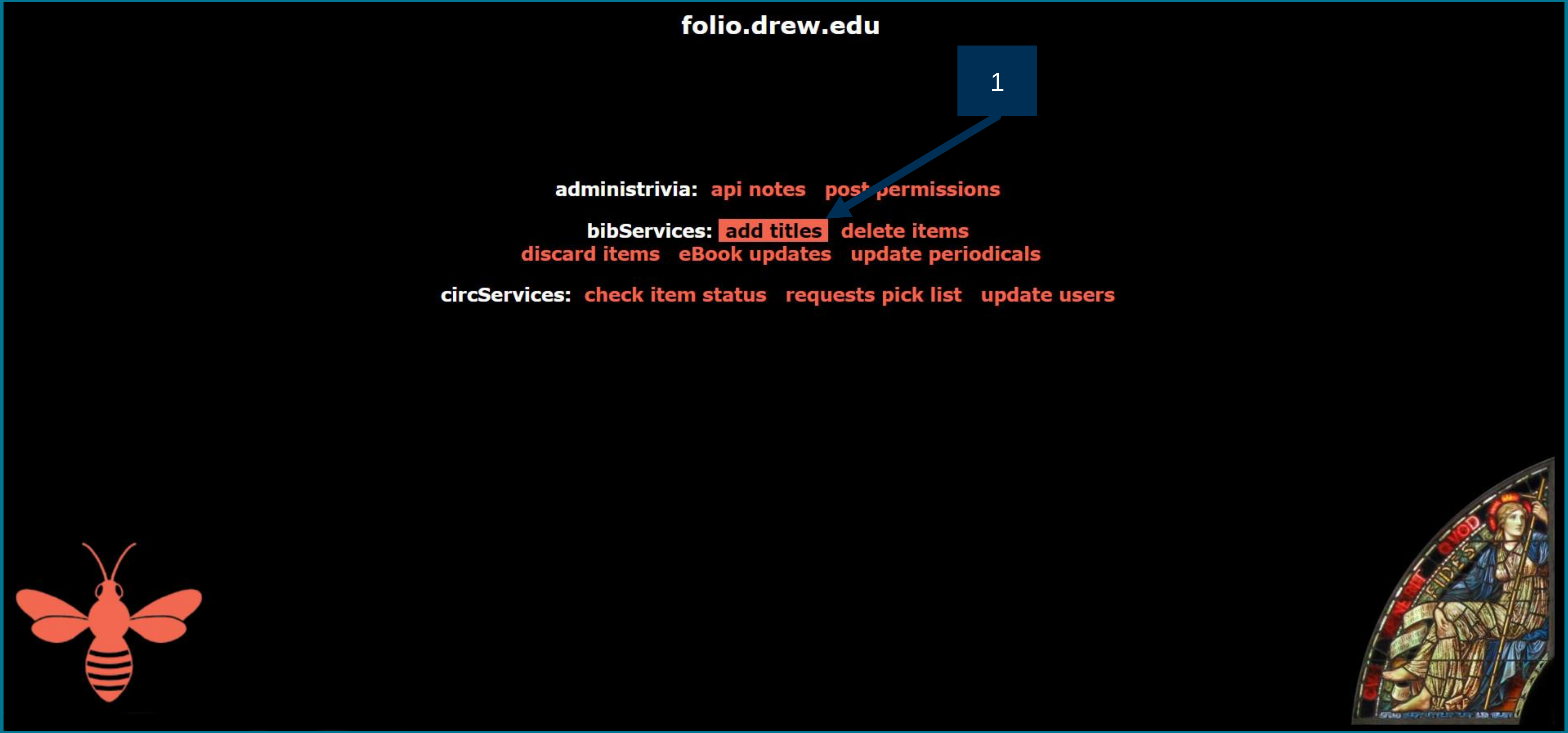
This web app boils the questions that need to be answered down to the minimum and makes educated guesses for most of those leaving little for the librarian to do other than scan a barcode.

via the UI

1. Apps
2. Data import
3. choose files
4. browse, select, Open
5. select Job profile
6. Actions
7. Run
8. Run
9. click File name
10. click "Created"
11. Add holdings
12. open Permanent location select box
13. select location
14. open call number type select box
15. select call number type
16. enter call number
17. Save and close
18. Add item
19. click on the barcode text box
20. scan the barcode
21. open material type select box
22. select material type
23. scroll down
24. open permanent loan type select box
25. select loan type
26. Save and close

via the web app

1. add titles
- select location, maybe
2. Choose file
3. browse, select, Open
4. next
- double check the
  - call number
  - call number type
  - loan type
  - Location
  - material type
5. Scan the barcode
- make it so

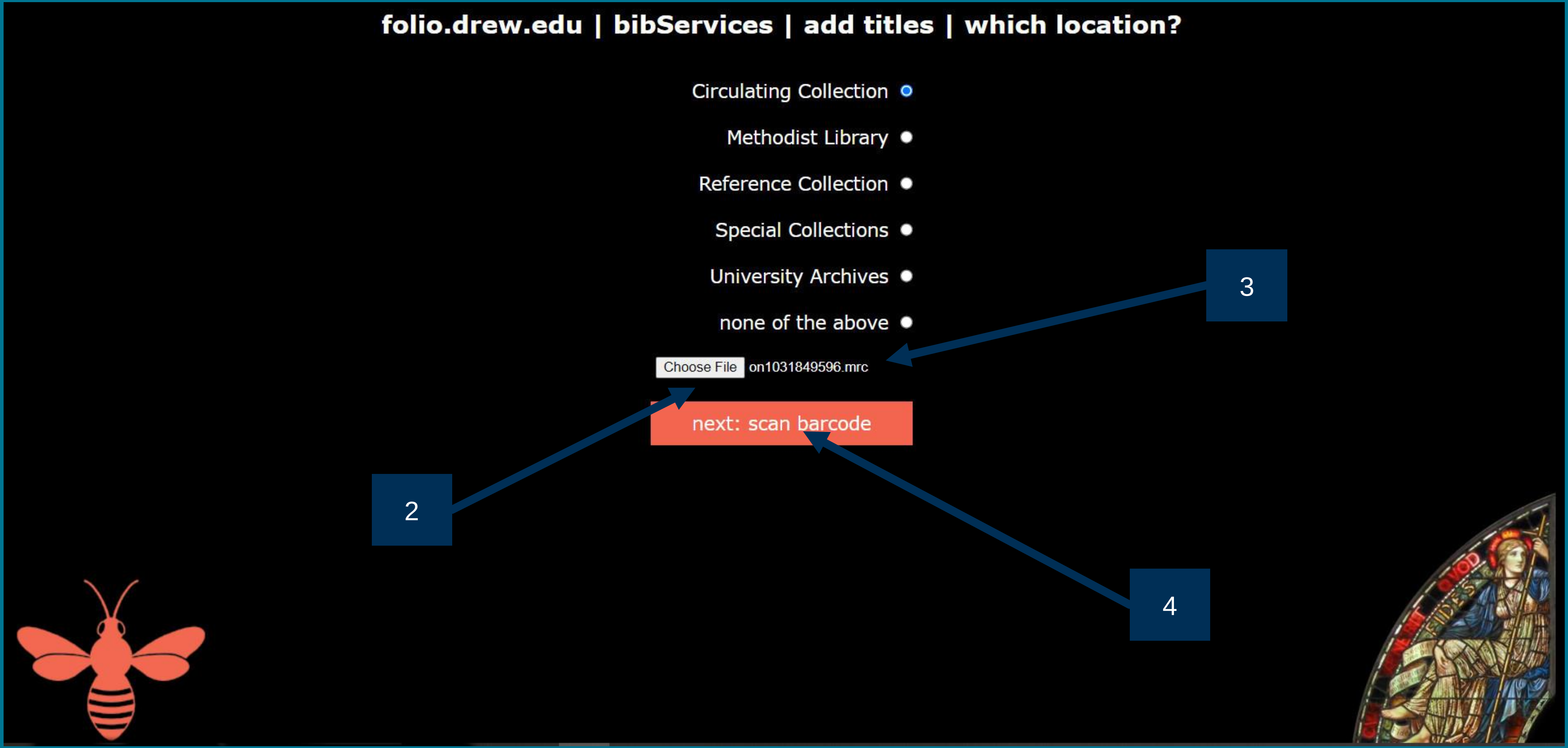


folio.drew.edu

administrivia: api notes post permissions

bibServices: add titles delete items discard items eBook updates update periodicals

circServices: check item status requests pick list update users



folio.drew.edu | bibServices | add titles | which location?

Circulating Collection [✓]

Methodist Library

Reference Collection

Special Collections

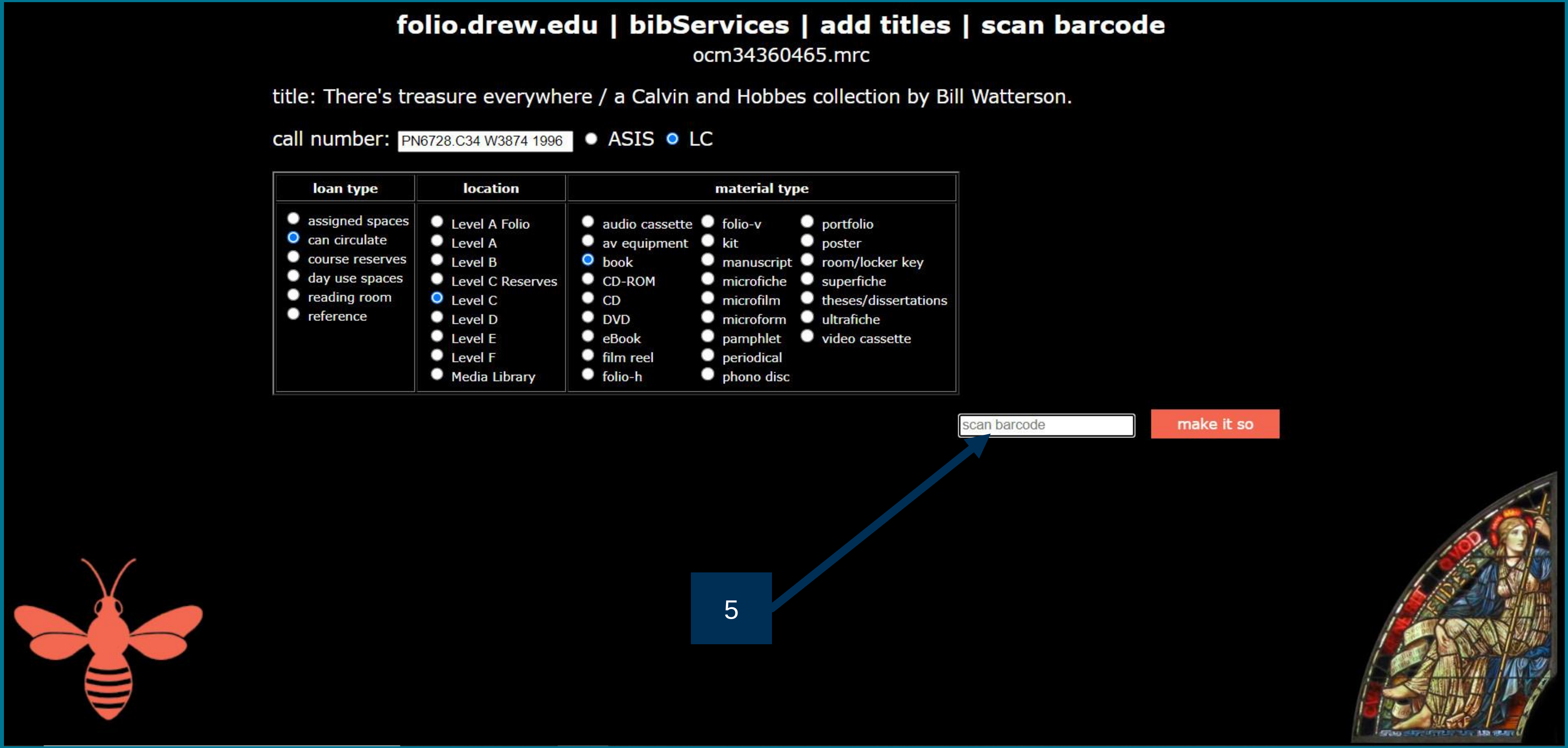
University Archives

none of the above

Choose File on1031849596.mrc

next: scan barcode

EBSCO USER GROUP



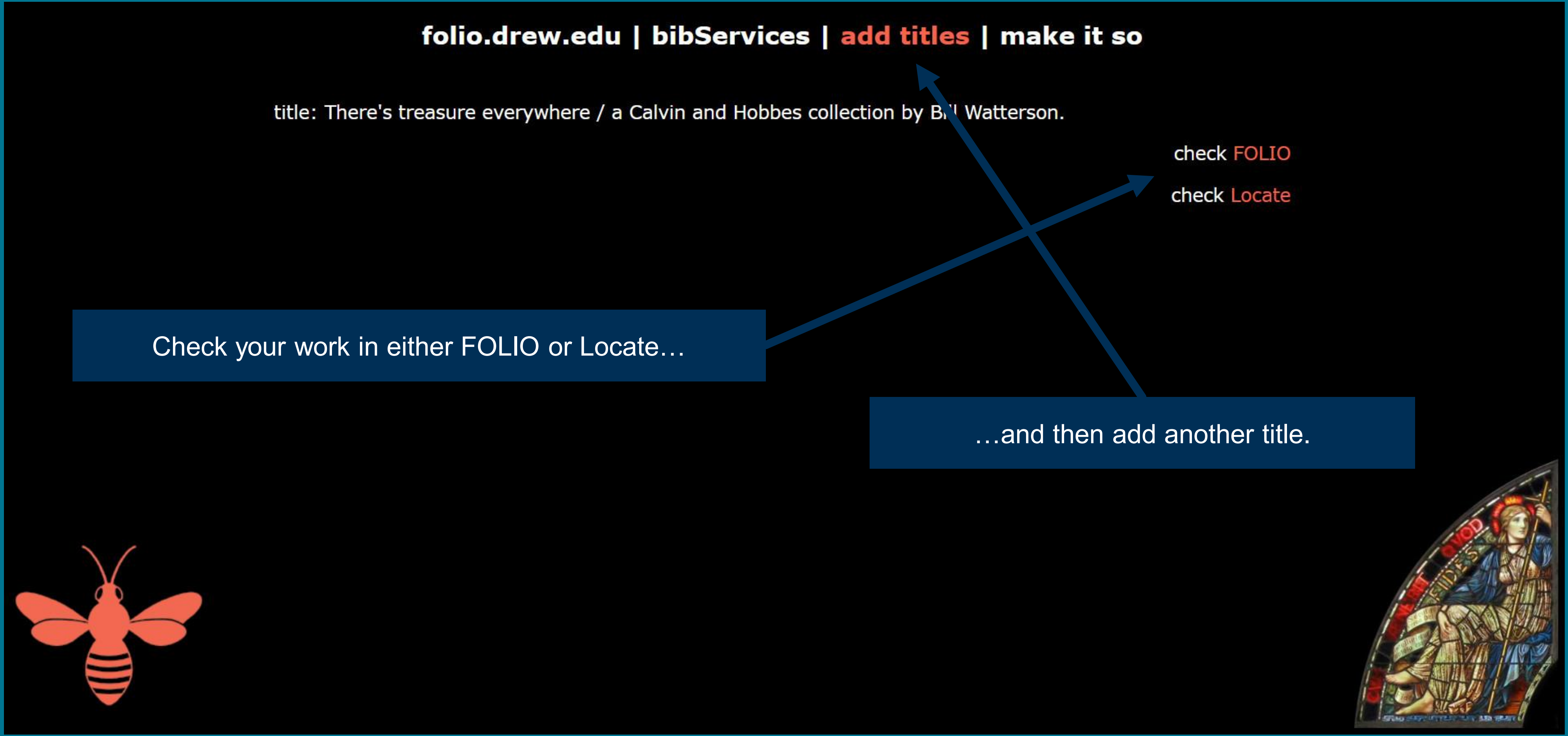
folio.drew.edu | bibServices | add titles | scan barcode

ocm34360465.mrc

title: There's treasure everywhere / a Calvin and Hobbes collection by Bill Watterson.

call number: PN6728.C34 W3874 1996 ● ASIS [✓ LC]

loan type	location	material type		
assigned spaces	Level A Folio	audio cassette	folio-v	portfolio
[✓ can circulate]	Level A	av equipment	kit	poster
course reserves	Level B	[✓ book]	manuscript	room/locker key
day use spaces	Level C Reserves	CD-ROM	microfiche	superfiche
reading room	[✓ Level C]	CD	microfilm	theses/dissertations
reference	Level D	DVD	microform	ultrafiche
	Level E	eBook	pamphlet	video cassette
	Level F	film reel	periodical	
	Media Library	folio-h	phono disc	



folio.drew.edu | bibServices | add titles | make it so

title: There's treasure everywhere / a Calvin and Hobbes collection by Bill Watterson.

check FOLIO

check Locate

Check your work in either FOLIO or Locate…

…and then add another title.

how it works

- chooseFile.pl collects the $locationGroup from whichLocation.pl
- scanBarcode.pl displays the title and puts the call number that is in the 050 in the call number text box
- for the circulating collection there is logic that looks at the call number to determine what the location should be
- the call number type (LC), loan type (can circulate), and material type (book) are simple defaults
- makeItSo.pl saves the MARC as [username].mrc and then…

1. **POST -d '{"fileDefinitions":[{ "name":“[username].mrc" }]}' /data-import/uploadDefinitions**

   this creates an uploadDefinition which supplies an $uploadDefinitionId, $fileDefinitionsId, and $jobExecutionId

2. **POST -d @[username].mrc /data-import/uploadDefinitions/$uploadDefinitionId/files/$fileDefinitionsId**

   this actually uploads the MARC to the server

   * this requires **-H 'Content-type: application/octet-stream‘** rather than the usual **-H 'Content-type: application/json‘** *

how it works, continued

- With the help of regular expressions replace the metadata in the $uploadDefinition with the necessary jobProfileInfo including the id, name, and datatype

3. **POST -d '$uploadDefinitiion' /data-import/uploadDefinitions/$uploadDefinitionId/processFiles**

   this tells FOLIO to get the job, Create instance and SRS MARC Bib, done – but it takes a moment, so…

4. **GET /metadata-provider/jobLogEntries/$jobExecutionId**

   …repeats itself, and counts Mississippis, until the value of totalRecords is greater than zero. Once that's done the entry will supply the $sourceRecordId so that…

5. **GET /source-storage/records/$sourceRecordId**, which also requires counting Mississippis, can supply the $instanceId and…

how it works, continued

- …a simple $json can be assembled:

```
{
		"instanceId":"$instanceId",
		"callNumber":"$callNumber",
		"callNumberTypeId":"$callNumberTypeId",
		"permanentLocationId":"$locationId“
}
```

6. **POST -d '$json' /holdings-storage/holdings** then creates the holdingsRecord and supplies the $holdingsRecordId

EBSCO USER GROUP

how it works, continued

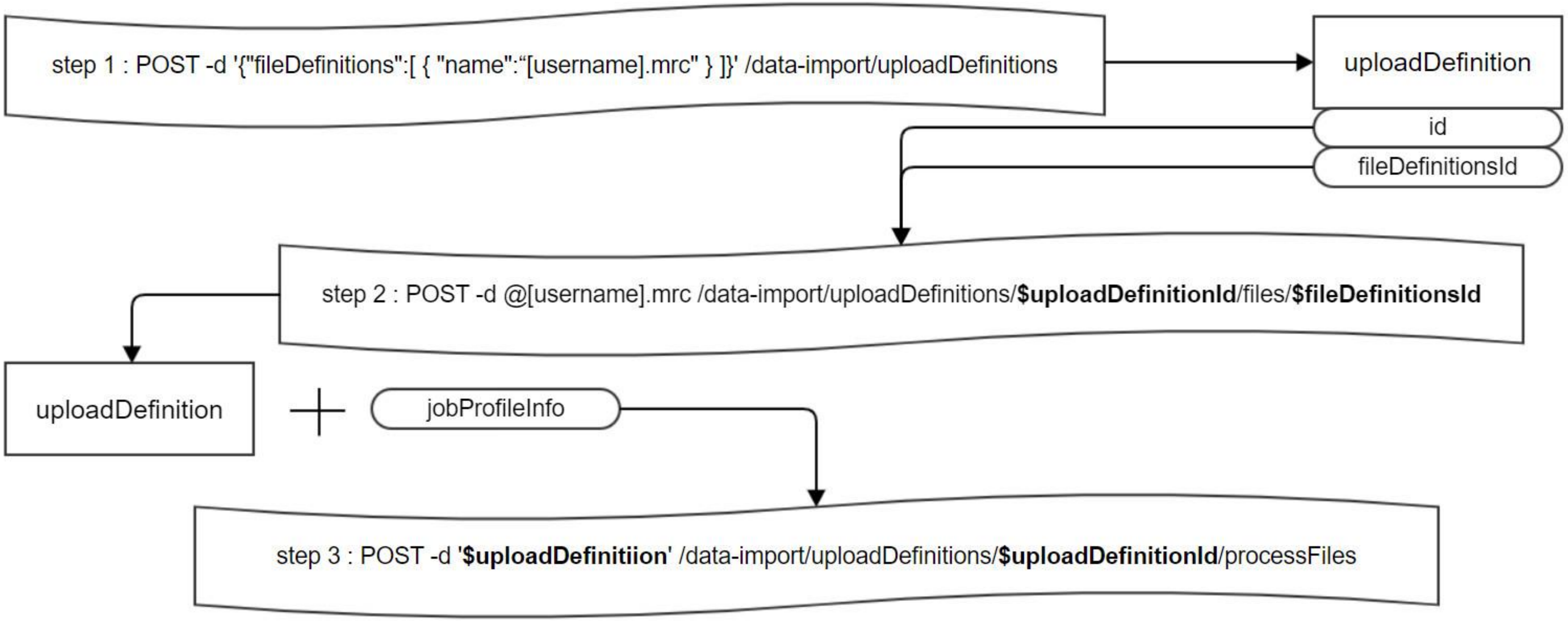
- another simple $json can then be assembled to create the item record:

```
{
        "holdingsRecordId":"$holdingsRecordId",
        "barcode":"$barcode",
        "materialTypeId":"$mtypeId",
        "permanentLoanTypeId":"$loanTypeId",
        "status" : {"name" : "Available"}
}
```

7. **POST -d '$json‘ /item-storage/items** then create’s the item and Bob’s your uncle.

- links into FOLIO and Locate are then provided so that you can check your work

how it works, continued

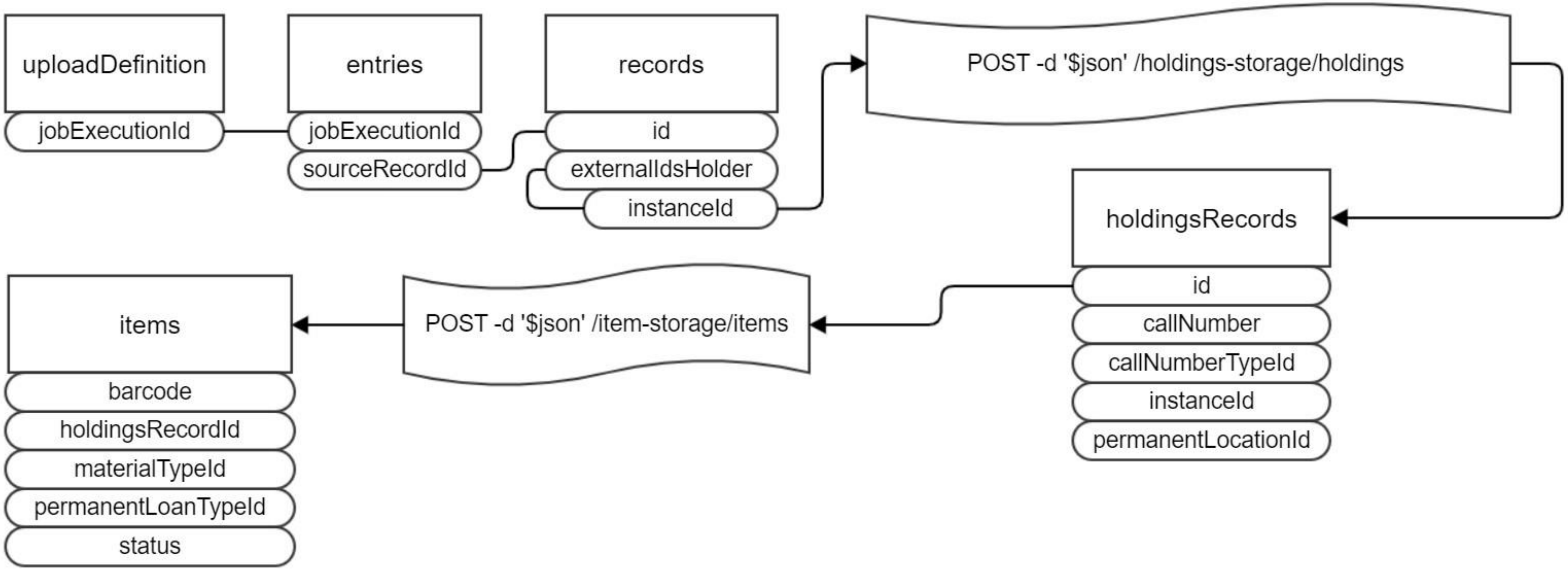


```
step 1 : POST -d '{"fileDefinitions":[ { "name":"[username].mrc" } ]}' /data-import/uploadDefinitions
```

step 2 : POST -d @[username].mrc /data-import/uploadDefinitions/**$uploadDefinitionId**/files/**$fileDefinitionsId**

step 3 : POST -d '**$uploadDefinitiion**' /data-import/uploadDefinitions/**$uploadDefinitionId**/processFiles

how it works, continued



comments? ideas? thoughts? questions?

Thank You

gdobson@drew.edu / folio.drew.edu