Data Migration -- Loans

- Decision points
- Step-by-step guide for using the business logic APIs.
- Step-by-step guide for using the storage APIs.
- Other considerations (gotchas)
- Related articles

Useful when migrating open loans from a legacy system to FOLIO. As of early 2020, two different migration techniques are being used, each with its pros and cons. One of these is to migrate open loans by submitting basic loan data to the appropriate business logic modules and then modifying due dates and other data as necessary. The other is to submit loan data to the appropriate storage modules. See below for the steps involved in each technique.

### Decision points

The following table attempts to present the trade-offs and decision points between the two techniques to assist migration staff in selecting a technique appropriate for their institution.

<table>
<thead>
<tr>
<th>Business Logic APIs</th>
<th>Storage APIs</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>The json structure for initial submission to /circulation/check-out-by-barcode is very simple</strong> (<a href="https://s3.amazonaws.com/foliodocs/api/mod-circulation/p/circulation.html#circulation_check_out_by_barcode_post">https://s3.amazonaws.com/foliodocs/api/mod-circulation/p/circulation.html#circulation_check_out_by_barcode_post</a>). There are only 3 required elements: itemBarcode, userBarcode, servicePointId. The loan record resulting from the submission to /circulation/check-out-by-barcode can be captured, modified to update dates, etc., and then submitted to /circulation/loans.</td>
<td><strong>The json structure for submission to /loan-storage/loans is much more complex</strong> (<a href="https://s3.amazonaws.com/foliodocs/api/mod-circulation-storage/p/loan-storage.html#loan_storage_loans__loanid__put">https://s3.amazonaws.com/foliodocs/api/mod-circulation-storage/p/loan-storage.html#loan_storage_loans__loanid__put</a>) but gives the programmer more control over the process. No secondary, or &quot;clean-up&quot; submission is necessary.</td>
</tr>
<tr>
<td>Submitting the json to /circulation/check-out-by-barcode automatically assigns the policies encoded in Settings/Circulation/Circulation rules.</td>
<td>Unless the tenant has decided to use default policies for migration, the programmer must determine the appropriate loanPolicyId, overdueFinePolicyId, and lostItemPolicyId and store them in the loan record before submission to /loan-storage/loans. Each of these policies can be determined by a call to the appropriate /circulation/rules/[policy type] API (e.g. /circulation/rules/loan-policy). The call must include the FOLIO item_type_id (material type), loan_type_id, patron_type_id, and location_id for the effective location for the patron and item associated with the loan.</td>
</tr>
<tr>
<td>Submitting the json to /circulation/check-out-by-barcode automatically creates all of the appropriate notices encoded in Settings/Circulation /Circulation rules, although initially the scheduled date for the notice corresponds to a current loan rather than a retrospective loan. When the updated loan record, with original loan date, current due date, and &quot;action&quot;: &quot;dueDateChanged&quot; is submitted to /circulation/loans, the system will delete the original notices and create new notices for the loans with the 'nextRunTime' set as appropriate for the updated due date.</td>
<td>The programmer must determine which notices are appropriate given the current state of the loan and create the json for them according to <a href="https://s3.amazonaws.com/foliodocs/api/mod-circulation-storage/p/scheduled-notice-storage.html#scheduled_notice_storage_scheduled_notices_post">https://s3.amazonaws.com/foliodocs/api/mod-circulation-storage/p/scheduled-notice-storage.html#scheduled_notice_storage_scheduled_notices_post</a>. For example, some items may require a courtesy notice and others may not. The resulting notice json is then posted to /scheduled-notice-storage/scheduled-notices.</td>
</tr>
</tbody>
</table>

All users who have items checked out must be loaded with "active": "true" because FOLIO will not process checkouts to inactive users. Once the migration process is complete, user records can be inactivated as appropriate.

Users who have items checked out must have a barcode because barcode is a required data element in the check-out-by-barcode json.

All items that have open loans must be loaded with status = 'Available' so that /circulation/check-out-by-barcode can process them. This will result in a status of 'Checked out'. As of early 2020 it is unknown whether this technique will ultimately result in the desired status for the item. For example, because "age to lost" processing is not fully implemented, it is not known whether a migrated loan that has passed its age to lost period will be assigned a "Lost" status.

### Step-by-step guide for using the business logic APIs.

- Migrate all users who have items checked out with "active": "true".
- Migrate items to inventory. If item is checked out, status must be set to 'Available' so that the /circulation/check-out-by-barcode API can process them.
- Create all appropriate loan policies, request policies, overdue fine policies, patron notice policies, lost item policies and circulation rules.
- Set service point calendar start dates to include the earliest possible loan date and latest possible due date according to loan policies.
• Turn off circulation notice timers by creating a new module descriptor and deployment descriptor for mod-circulation without timers, and enabling for the tenant. It is desirable to create and POST new module and deployment descriptors rather than updating the existing ones so that the sysadmin can use the Okapi tenant install API to switch the module for the tenant seamlessly, with no other side effects, once the migration is complete.

• Load loans:
  • POST json to /circulation/check-out-by-barcode API
  • capture the json for the resulting loan record, change loan date, due date, and set "action" : "dueDateChanged" in the loan json. Updating the "action" value will ensure that the system will delete the original notices and create new notices for the loans with the 'nextRunTime' set as appropriate for the updated due date. Can also change "renewalCount" if desired.
  • PUT the updated loan json to /circulation/loans.

• Delete patron action sessions from the database, which will keep "just checked out" notices from being sent out. This can be done directly in the database:

```
DELETE FROM tenantId_mod_circulation_storage.patron_action_session
```

OR via a series of API calls

• Get the Patron action session for the loan

```
GET {{baseUrl}}/patron-action-session-storage/patron-action-sessions?query=(loanId="LOAN_ID")
```

• Delete the session

```
DELETE {{baseUrl}}/patron-action-session-storage/patron-action-sessions/SESSION_ID
```

• Delete all scheduled notices with date < today from the database. This will keep the system from sending out notices for loans for which the notices were presumably sent by the legacy system. This can be done directly in the database:

```
DELETE FROM tenantId_mod_circulation_storage.scheduled_notice WHERE (jsonb->>'nextRunTime')::timestamp < NOW()
```

OR via a series of API calls:

• Capture the patronID from the resulting loan record
• Get the scheduled notices with a next runtime less than today

```
GET {{baseUrl}}/scheduled-notice-storage/scheduled-notices?query=(nextRunTime<"YYYY-MM-DD*" and loanId="LOAN_ID")
```

• Capture the ids of the scheduled notices and make DELETE requests for each of them:

```
DELETE {{baseUrl}}/scheduled-notice-storage/scheduled-notices/NOTICE_ID
```

• Reenable timers by enabling mod-circulation with timers for tenant
• Update user records to "active": "false" if appropriate.

### Step-by-step guide for using the storage APIs.

• Migrate users. Users may be set to "active": "true" or "active": "false" as appropriate.
• Migrate items. Item status may be set as appropriate, e.g. 'Checked out', 'Available', etc.
• Create all appropriate loan policies, request policies, overdue fine policies, patron notice policies, lost item policies and circulation rules.
• Set service point calendar start dates to include the earliest possible loan date and latest possible due date according to loan policies.
• Create json according to /loan-storage/loans and POST to that endpoint.
• Create json for patron notices as appropriate given the state of the loan and POST to /scheduled-notice-storage/scheduled-notices.

### Other considerations (gotchas)

• Make sure you consider daylight saving time for importing loan information - depending on when the loan was created, the time offset may need to be different to get it to display correctly in the local time zone in FOLIO.
Related articles

- Data Migration -- Loans