LASER/ERM Integration

Laser/ERM integration is a collection of features aimed at allowing a LASER site to synchronize their subscriptions (Content Lists) and licenses with the FOLIO ERM Agreements and Licenses apps. The integration works via a new backed module: mod-remote-sync (https://github.com/folio-org/mod-remote-sync) and related UI: ui-remote-sync (https://github.com/folio-org/ui-remote-sync). Taken together, remote-sync is a general purpose scriptable ETL tool for FOLIO into which arbitrary configurations can be loaded and used to keep FOLIO in sync with a remote data source. Whilst remote-sync itself is intended to be general and reusable, the configuration for LASER is solution specific and can be found here (https://raw.githubusercontent.com/k-int/folio-sync-tools/main/laser/laser_registry.json).

Installing the remote-sync configuration for LASER.

Configuring

We refer to bundles of configuration such as the laser config as a "Registry" - you can upload a registry from settings Remote Sync by adding the URL of the RAW config file and pressing the save button.

If all goes well, the response looks like this (Currently)

Because a registry contains executable code with access to your folio tenant data each bundle of code must be signed by a key. The signature is checked when loading the bundle to ensure that the code has not been tampered with. The list of keys which can sign registries is currently controlled by k-int.

If all goes well, the main remote-sync screen should now look like this:
On the left side, the record source is listed. Boxes in this column represent the actual raw records which will be / have been collected from a remote source. Remote sync runs on a cron schedule which is invoked every 60 minutes. If a previous extract job has not yet completed, the task is skipped. This means remote-sync will try to synchronize with a remote source once per hour. Because some sources (LASER) do not provide a good cursor mechanism the only way to tell if a record has changed is to download the content and compare the current version with the previous one. That is exactly the process which takes place in this box for LASER. The main aim of the first column is to take a datasource with heterogeneous cursor semantics and turn it into a record stream which can be enumerated. The boxes on the left hand side correspond to the “source” record types in the registry.

The High Level remote-sync workflow

Once in each sync cycle (Each hour) the boxes in the second column will look for any records in their supplying streams which have changed. For an OAI source this would be simple, for sources such as LASER this is computationally expensive. remote-sync will invoke any processes which are interested on events

Specific LASER Semantics

The following sections essentially describe the code found in

- https://github.com/k-int/folio-sync-tools/blob/main/laser/ProcessLaserLicense.groovy and
- But both procedures need to be interpreted in terms of the config found in https://github.com/k-int/folio-sync-tools/blob/main/laser/laser_registry.json

LASER LICENSE Procedures

License Import Procedure - Control

For each incoming license (New licenses, or newly updated records)

1. If we have NOT seen this record before
   a. Attempt to look up feedback where the user indicates if we should import/ignore/map
   b. If "Create" feedback located - proceed to FOLIO license creation using the createLicense
   c. If "Ignore" feedback located - ignore the record
   d. If "Map" feedback is located - proceed to map this incoming license to an existing FOLIO license using the updateLicense procedure
2. If we HAVE seen this record before
   a. Proceed to updateLicense procedure

License Import - createLicense

1. Create a license with
   a. name = laser_record.reference
   b. description = “Synchronized from LAS:eR license ${laser_record?.reference}/${laser_record?.globalUID} on ${new Date()}”
   c. type = laser_record.calculatedType ?: laser_record.instanceOf.calculatedType ?: ‘NO TYPE’
   d. customProperties = customPropertiesProcedure
e. status = a value from the configuration which maps incoming laser statuses to FOLIO statuses
f. localReference = laser_record.globalUID
g. startDate = laser_record.startDate
h. endDate = laser_record.endDate

2. Post that to FOLIO
3. Store the resultant FOLIO ID so we know in the future which record FOLIO License maps to this LASER license

License Import - updateLicense

1. locate the FOLIO license with the ID which is mapped to this laser license (Stored in step 3 of createLicense, or in the mapping for "Map" operations)
2. overwrite values in the FOLIO license
   a. folio_license.name = laser_record?.reference
   b. folio_license.description = "Updated from LAS:eR license ${laser_record?.reference}/${laser_record?.globalUID} on ${new Date()}"
   c. folio_license.startDate = laser_record?.startDate;
   d. folio_license.endDate = laser_record?.endDate;
   e. folio_license.endDate = laser_record?.endDate;
3. Post the updated license back to FOLIO

customPropertiesProcedure

1. for each property in the incoming LASER license
   a. Look up the property in the mapping configuration
      i. If Found
         1. Look at the TYPE defined in the mapping configuration
            a. Text:
               i. Join all the LASER notes together and add them to the custprop note field
               ii. Add the value of the custom property as a TEXT field
               iii. If there is NO-Value set the value to "No-Value"
            b. Date
               i. Join all the notes as for text
               ii. Store the value in the value field
            c. Refdata
               i. Join notes
               ii. Look up the value in the mappings config
               iii. If a mapped value is found, set the value
               iv. If not, ignore the value.
      ii. If Not Found
         1. Ignore

LASER Subscription procedures

For each incoming subscription

1. If the sync titles setting is yes
   a. Generate the FOLIO Package import JSON for the subscription
   b. upsert (Insert or update) the package definition and remember the package details (UUID)
2. upsert (Insert or update) the subscription

upsertPackage procedure

We defer entirely to the agreements package ingest routine here - there is no special processing in remote-sync. Agreements uses package reference to decide if it should import or update a package so we hand the whole bundle off to the /erm/packages/import endpoint

upsertSubscription procedure

1. check to see if we have seen this incoming sub before.
   a. If we have - attempt the agreementUpdate procedure
   b. otherwise - see if we have feedback about creating, ignoring or mapping this subscription
      i. create - create a new agreement and store the ID so we can map it in the future
      ii. ignore - do nothing
      iii. map - retrieve the mapped ID from the user feedback and apply the agreementUpdate procedure

agreementCreate procedure
1. Create the agreement lines
   a. use the package ID created in the upsertPackage procedure to connect the agreementLine to the package
   b. set the AL note to "LASER:$(subscription.globalUID)"
   c. Set activeFrom to subscription.startDate
   d. set activeTo to subscription.endDate
2. apply buildPeriods procedure
3. Set the linked license to any license referenced by this subscription
4. Post the agreement
   a. [
      name: subscription.name,
      agreementStatus: statusString,
      reasonForClosure: reasonForClosure,
      description: "Created by remote-sync from LAS:ER on $(new Date())",
      localReference: subscription.globalUID,
      periods: periods,
      linkedLicenses: linked_licenses,
      items: items,
      customProperties: processSubscriptionProperties(rms, [], subscription, local_context),
      isPerpetual: subscription.hasPerpetualAccess
   ]

agreementUpdate procedure

1. lookUp any existing controlling license
2. If the existing (FOLIO) license data is different to the new (LASER) license data unlink the existing controlling license and link the new controlling license
3. update periods
   a. for each period in the FOLIO agreement. If the FOLIO period DOES NOT overlap with the period from the LASER agreement
      i. Add the agreement period as a new period to the FOLIO agreement
   b. if it DOES overlap
      i. We cannot continue, ask the user to rectify the dates
4. Check that there is an AL for the package attached to this sub - if not - add a new AL for the package (Title list changed at laser effectively)
5. post updated subscription information

buildPeriods procedure

1. clear any existing periods from the FOLIO record
2. replace that period with a new period as per
3. Map newPeriod = [
   startDate: subscription.startDate,
   endDate: subscription.endDate
]