Building Report Prototypes and Queries

Prioritize Reports and Assign Work

Report prioritization involves members of the Reporting SIG and their institutions. Initial priorities were included by institutions in the Reporting SIG Master Spreadsheet. Since transferring all reports to JIRA, institutions have been maintain rankings in JIRA instead. Most reports have been grouped into clusters, and the cluster issues are being ranked by institutions. The Reporting SIG functional area teams then coordinates prioritization of the clusters and assigning work to team members.

Write Report Specification

Create Report Wiki Page

- review report details, including data elements and report requirements, using the Reporting SIG Master Spreadsheet and the corresponding JIRA issue for the selected report
- create a name for the report and one or more queries that will be required for the report (see tips at Naming Conventions, but also try to mimic language commonly used for this report)
- create a wiki page for this report, using report name, prefaced by the functional area team
  - start on this page
  - using the "Create" button, create a new page – this should ensure that the new page is nested under "Building Report Prototypes"
  - for the title, start with the functional area team (e.g., MM, RA, RM), then a hyphen, then the human readable name of the report
  - save the page and check that it appears in the list under "Building Report Prototypes". If not, you can click on the page's menu and select "Move" to choose a new parent page.
- include a link to the JIRA issue for the report
- toward the beginning of the page, explain the purpose of this report
- include report name and names of all queries and describe any difference between queries
- add narrative text as needed, explaining the steps for the developers

Create JIRA Issue for Prototyping Work

- When you have a stub for the prototype on the wiki, contact Angela Zoss to have a new prototype issue created in the LDP-QURY project. The JIRA issue will include a link to the wiki page and will be connected to the JIRA issue for the cluster or report being prototyped.
- Angela will assign the issue to you when it has been created so you can keep track of all of your tasks. The workflow status will be changed to "In Development."

Identify Data Elements

- review report details on the Reporting SIG Master Spreadsheet and the corresponding JIRA issue
- Note: if working on a report cluster, or if the report has variations listed, make sure to identify data elements required across all queries
- for any fields mentioned in the issue or present in a sample report, locate an appropriate match in the FOLIO source data
  - go to https://dev.folio.org/reference/api/
  - scan the modules to find the general area that is most likely to include the data element
  - **use only "-storage" modules** (and/or avoid "business logic" or "bl" modules)
  - when you find a likely module, click on "View 1" or "View 2" for the description in the API reference documentation
  - In the View pages, you will see a list of paths to the endpoints underneath that module. For example, under the "loan-storage" module, you will see several endpoints: loan-storage/loans, loan-storage/loans/{loanId}, etc.
  - For the endpoints that contain data we'll be using in the LDP, you should see two very similar paths. One has nothing at the end (e.g., "loan-storage/loans") while the second should have an identifier at the end of the path (e.g., "loan-storage/loans/{loanId}"). You want to focus on the second path, the one with the identifier in curly brackets at the end.
  - Click on either the "GET" or the "PUT" link attached with that second endpoint path. If you use "GET", you will find the data element documentation under "Response." If you use "PUT", it will be under "Request."
  - find the correct element, and note the full path to the data element
  - if you have trouble understanding the data elements:
    - create a JIRA issue linked to **FOLIO-1551** - Getting issue details... STATUS for each data element in the FOLIO source data that is missing documentation or needs additional documentation; see examples in the "relates to" section under this epic

        - David Crossley has created a guide for FOLIO Developers to use to write FOLIO attribute documentation at https://dev.folio.org/guides/describe-schema/
        - for information from POs on data elements, you may refer to : https://docs.google.com/spreadsheets/d/1B924D2JaFUc6dqgItyKoOFdhn36lxwT6jaYmKpyk2Ws/edit#gid=1697459676
        - we are also building some Reporting SIG data documentation with helpful links and details
    - a note about the data elements in FOLIO
        - the FOLIO documentation should include a "type" for each data element
        - if the type is "array", that means that FOLIO is storing a list of items in this data element. When "type" is "array", you should also see "items", which will tell you what kind of element is being stored in the array.
        - if the type is "object", that means that FOLIO is storing a JavaScript object, which is like a dictionary or a list of key-value pairs. When "type" is "object", you should also see "properties", which will list all of the keys that might be included in the object.
        - See attached summary for more detail.
        - when you include FOLIO arrays and objects in a project, make sure you document the full path to the item you want and include notes about what (if any) processing might be needed, especially for arrays
- Once you have matched up the data elements that are visible in the final report, do a final pass through the FOLIO documentation to make sure that you also include any data elements that will be required for joining tables together. These are likely identifier elements. Even if you want just the name to appear in the report, please locate and list all required linking identifiers as well.
- On the report wiki page, create a table that lists all required data elements, their path in the FOLIO modules, and a description of how that field will be used (including whether or not the field should be displayed in the final report). Some find it helpful to group data elements by the FOLIO interface they come from, including adding an empty row that shows the name of the interface.
- Optional: for especially complicated reports, it might be nice to include a relationship diagram that helps query builders make the correct connections between tables

EXAMPLE TABLE

Folio Attribute (Module, Path, Attribute)	Folio Data Element Description	Parameters/Query
(mod-users:) groups/**id**	A UUID identifying this (patron) group	sort and count results by (patron group) id
(mod-users:) groups /**group**	The unique name of this (patron) group	
(mod-users:) users/**id**	A globally unique (UUID) identifier for the user	
(mod-users:) users/**patronGroup**	A UUID corresponding to the group the user belongs to; this is a reference to (mod-users:) groups/**id**; this is a foreign key to the id attribute under groups	
(mod-circulation-storage:/loan-storage/loans)**id**	Unique ID (generated UUID) of the loan; represents the loan transaction; storage module; we get data from the storage module	count loan transactions
(mod-circulation-storage: /loan-storage/loans) **userId**	foreign key; ID of the patron the item was lent to. Required for open loans, not required for closed loans (for anonymization).	
(mod-circulation-storage: /loan-storage/loans) **loanDate**	Date time when the loan began	filter loan transaction ids by date range
(mod-circulation: /circulation /loans)**id**	Unique ID (generated UUID) of the loan; business logic module; called "loanid" in the reporting database, will be marked as a foreign key	
(mod-circulation: /circulation /loans)**location**	The effective location of the item; nested under "items"	

Include Sample of Report Output

- After mapping out the required data elements, include a sample of the desired report output. The query writers will be trying to match this output.
    - The JIRA issue may have an attachment, or the master spreadsheet may have a link - reproduce one or both here
    - Alternately/additionally, can create a table in the wiki page that simulates the output
- Note: if report has query variations listed, include separate samples for each query that needs to be developed
- Make sure it is clear if a filter or aggregation has been applied. That is, consider adding a dummy column that lists, e.g., the date range filter that was used for this report.

Placeholder: review all included data elements to see if they have proper documentation in LDP data dictionary

Review Specification with Reporting SIG, SMEs

- After the draft of the prototype is done, the workflow status in JIRA should shift to "In SIG review." If you cannot make this change yourself, contact Angela Zoss or have your team lead contact her.
- Have Reporting SIG review the specification, starting with the functional area team. Reporting SIG should make sure the list of elements make sense and that any remaining questions about report purpose and output are noted.
- After Reporting SIG has reviewed, make any requested changes. At this point, the workflow status will shift to "In SME review."
- The prototype should then be shared with the FOLIO SIG(s) interested in this report. Raise any remaining questions, and make sure that the purpose of the report has been understood accurately. Have them especially review the proposed output to make sure the correct columns are included (and excluded) in the final report.
- After the FOLIO SIGs have reviewed, make any requested changes. At this point, the workflow status will shift to "closed."
- When all approve the specification and the prototype is completed, the prototyper should work with the functional area team lead to begin work on the query development.

Write Query

Create a LDPQURY Issue for Your Work on This Query

- When you are assigned a query to work on, contact Angela Zoss and she will create a JIRA issue in the LDPQURY project
- Angela will assign the issue to you when it has been created so you can keep track of all of your tasks. The workflow status will be changed to "In Development."
- At this time, Angela will also setup a subdirectory in the sql folder of the ldp-analytics repository. You can begin building and versioning your query in GitHub right away, or you can develop the query locally and push it to GitHub when it is ready for review.

Build SQL Query

- Create new folder in ldp-analytics/sql to store sql, using the name of the query specified in the wiki page
    - If you have any trouble, contact one of the maintainers of ldp-analytics
- Create a file with the query name and the extension ".sql"
- Follow SQL conventions specified in LDP documentation
- Identify tables and data elements in the LDP. If you have any questions about the mapping between FOLIO interfaces and LDP tables, reach out to other query developers or Nassib.
- Let developers know ASAP if we need additional test data for these data elements, or if any needed tables are missing from the LDP database. Please use the FOLIO Issue Tracker for these purposes, by creating an issue in the "Library Data Platform (LDP)" project. Set the issue type to "New Feature", and fill in the summary and description fields. Please do not set any other fields in the issue.
- Additional tips for SQL:
    - Use all-caps for keywords (e.g., "SELECT", "FROM")
    - Start new lines for new keywords, items in long lists
    - Indent lines with 4 spaces
    - Review previous queries for other style conventions
    - Review Angela's recorded presentation on building an SQL query (name of the file on the Google drive is *GMT20190920-135741_RPWG--Quer_1760x900.mp4*)
    - Preferred: use Nassib's pgformatter configuration file to format the SQL automatically. (Install pgformatter and set up the configuration file. Then, after creating a .sql file, run the formatter over the SQL using the command `pg_format`. For example: `pg_format query.sql -o output.sql`. If you have a backup of the query file, you can tell pg_format to overwrite the original query "in place" with the formatted output: `pg_format -i query.sql`.)

Test Query

- Set up a connection to the test LDP (see FOLIO Reporting Reference Environment for connection details)
- Log into the LDP using your reporting tool of choice
- Execute the query
- Review results and revise query as needed

Push Query Back to GitHub

- Your first time working on new queries that will go up on GitHub, you'll need to create a fork of the ldp-analytics repository.
- See Using GitHub to develop report queries for folio-analytics
- If you use GitHub throughout your development process, you should commit and push your file edits to GitHub periodically
- Once the query is complete, you can submit a "pull request" to have the new query merged into the ldp-analytics repository

Review Query with Reporting SIG, SMEs

- After the draft of the query is done, please notify Angela Zoss. The workflow status in JIRA will shift to "In Code Review," and Angela will review the query for basic syntax and logic concerns. She will also test the query agains the Amazon Redshift test instance of the LDP. She may be in touch with questions, or she may simply make changes and let you know what happened.
- When code review has been completed, the workflow status will shift to "In SIG review."
- Have Reporting SIG review the specification, starting with the functional area team. Reporting SIG should make sure the query logic makes sense, the correct fields are being used, the output looks reasonable, and that any remaining questions about report purpose and output are noted.
- After Reporting SIG has reviewed, the query developer (and/or Angela) will make any requested changes. At this point, the workflow status will shift to "In SME review."
- The query should then be shared with the FOLIO SIG(s) interested in this report. Raise any remaining questions, and make sure that the query functionality (e.g., requested filters/aggregations) and output look appropriate.
- After the FOLIO SIGs have reviewed, make any requested changes. At this point, the workflow status will shift to "In Automated Testing." Angela Zosswill generate test files for the query.
- After the test files are generated, the workflow status will shift to "In QA." Reporting SIG members will need to volunteer to test the query on different machines.

Finalize Documentation

- When testing is complete, query documentation needs to be updated.
    - The original query developer will work with Angela to update the original prototype in the wiki and move the final version to under the XX-Completed Prototypes Archive page.
    - The purpose of the report and the updated sample table should be added to the README file for the query on GitHub. (For an example, see the "Circulation Detail Report" query.) The README should have a basic description of the query, including its purpose and any parameters or aggregations, as well as a sample table of results.

- Create a pull request to update the master repository with the new README.
- At this point, the JIRA issue for the query can be closed.