Preparing Input Data and Running an ETL Process

Introduction

SWIFT ETL commands require that all data be entered into SWIFT templates. One SWIFT template must be used for each resource type to be ETLd. Typically, the primary resource type you are ETLing i (e.g. Plasmid) will require several secondary resource types (e.g. Person, Journal Article, ...) to be fully described in eagle-i -- the secondary resources will be linked from the primary resources.

Entering data correctly into SWIFT templates is key for the ETL process to succeed without creating duplicates or non-conforming data. You may enter data into SWIFT templates in the following ways:

- Manually add data, row by row.
- If data exists electronically: obtain a data dump, pre-process it (see below) and copy individual columns into the SWIFT template.

Understanding the original data and pre-processing it

Data that exists electronically will typically be stored in a relational database and accessible via a database dump, or accessible through an API (e.g. in JSON format). It is usually necessary to perform a few transformations on the original data in order to fit it into a SWIFT template. This step is highly dependent on the nature of the original data, and hence the procedures will need to be developed on a case by case basis. In mapping the data from its original schema to the eagle-i ontology, the following scenarios may be encountered:

- Original data may need to be split into multiple eagle-i resource types
- For a given type, there may be a one-to-one correspondence between the original field (e.g. column) and a property in the eagle-i ontology
- For a given type, a field in the original data may need to be split into multiple eagle-i ontology properties
- For a given type, fields in the original data may need to be combined into one eagle-i ontology property
- A controlled vocabulary field in the original data may need to be mapped to a term in one of eagle-i's referenced taxonomies

We usually write ad-hoc scripts that perform such transformations on the original data before copy-pasting individual columns into a SWIFT template. For controlled vocabulary fields, we produce mapping tables with the help of our domain experts and use them during these pre-processing steps.

Generating the templates

SWIFT templates and maps need to be generated using the toolkit version that corresponds to your eagle-i repository version (this is very important because ontology versions matter - if versions don't match, you might end up with non-conforming or incomplete resource descriptions).

When ETLing a primary type, there are usually resources of other types that are related to it (e.g People, Organizations, Publications). It is necessary to enter information for these related types in a separate template. For example, when ETLing a Monoclonal Antibody, you'll have separate files for related Hybridoma Cell Lines, People and Publications.

Use the `generate-inputs` command described in the SWIFT toolkit guide to generate the following templates:

- A template for your primary resource type - use the most specific type possible (e.g. Monoclonal Antibody and not Reagent).
- Templates for all the secondary resources you need - consult the eagle-i ontology browser page of your primary resource type to understand what types can be linked.

Do not modify the directories that are created by this command.

Adding your data to SWIFT templates

Create a directory dedicated to your data files, and subdirectories for each resource type. Copy the templates you will use to the appropriate subdirectory - note that more than one file of a given resource type may be used (e.g. you could have a file per lab if you're ETLing multiple labs) We recommend that before adding resource data to a template, you rename the file such that the name is meaningful, since all resources ETLd from a file will be tagged with that file's name, We've found it useful to use a name that reflects the date of the ETL , e.g. `20150627-NYSCF-human-study.xls`.

SWIFT Templates include different kinds of columns:

- Plain text columns - you may enter any text. Do not include semi-colons, as they are used as field separator (more on this below).
- Resource columns - represent secondary, linked resources.
    - A value that matches the label of a resource in the repository, in a secondary file, or a new resource is expected. If you know the resource's URI, you may enter it here and thus avoid a look-up operation during ETL.
    - Resource columns are followed by Resource type columns; sometimes the resource type column is omitted if there are no possible values (i.e. the resource type has no subclasses) - if the column exists, it must be used.
- Referenced taxonomies - a value that matches an eagle-i ontology term or synonym is expected - if you know the term URI, you may enter it here and thus avoid a look-up operation during ETL.
- The first two columns (hidden) of a template are reserved for metadata. Please do not modify them.

Consult the tooltips in the header rows for more information about what is expected.


It is highly recommended you delete any columns from the template that are not needed. This eliminates potential confusion and makes the sheets much less unwieldy.

Rules and guidelines

- You must always enter the Organization to which the resource is associated, either by name or by URI


It is best to use the Sweet to add the Organization (e.g. lab) to which these resources are associated, and then reference this name or URI in the files.

- References in the primary file, either to resources represented in a secondary file or resources in the repository**, need to use the exact label (ignoring case) for the correct linkage to occur**.
- If there is more than one value for a given column, enter values separated by ; (semicolon). Conversely, check your input file for the presence of ; in values that are not meant to be split and substitute for a different character.
- Every resource (primary or secondary) needs to have a name and a type as a minimum. For simplification, the type column is omitted if there are no possible subclasses (e.g. Person, Human Subject). If the template has a type column, you **must** enter a value.
- **Use URIs instead of labels with caution, as all checks are also by-passed** (in particular, the ETL process does not verify that the resource exists in the repository or that the term exists in the ontology, so you may end up with non-conforming data).
- Generate ETL templates for the most specific type needed, i.e. Antibody instead of parent type Reagent, Core Facility instead of parent type Organization, Biological process phenotype instead of parent type Phenotype, Journal Article instead of parent type Document, etc. This is because the properties generated for each of these more sub-types will be different than those generated for the root class.


We have found that the URI option is most useful when all values in the column are the same - otherwise it makes visual inspection of the data more difficult.

Special rules for generic organisms

Generic organisms are instances of an Organism subclass that are meant to represent that subclass as a whole, when no specific instances are needed in the data - for example the generic organism https://global.eagle-i.net/i/Mus_musculus represents the Organism subclass http://purl.obolibrary.org/obo/NCBITaxon_10090. Generic organisms typically reside in eagle-i's Centrally Curated Resources.
In your template, if a column can accept an Organism value, you may use a specific instance or a generic instance. If you are using a generic instance, enter the label (e.g. Mus musculus) in both name and type columns.

Special rules for embedded instances

Embedded resources are instances of Embedded Classes (see the eagle-i ontology browser). Embedded resources only make sense as part of another "main" resource and as such they can only be created or updated as part of a creation or update operation for that resource. Examples of embedded classes are Construct Inserts (embedded in Constructs) or Diagnosis (embedded in Human Subjects).

In the context of ETL and SWIFT templates, embedded resources are a special case of linked resources. If there is only one embedded resource in a given resource, it is possible to enter its information directly in the SWIFT template row for the main resource. If there are more than one embedded resources in a given resource, the following procedure must be used:

- In the template for the main resource, enter a list of semicolon-separated labels for the embedded resources, and fill the type column if it exists (only one value is required, **not** a semicolon-separated list). This will result in the creation of "empty" embedded resources
- ETL this file (see below)
- In order to ETL the rest of the properties for the embedded resources:
    - Generate a template/map for the embedded resource class
        - Templates generated for embedded classes will have two additional columns: *Main Resource Name* and *Main Resource Type*.

    - Fill the template with the information for the embedded resource (one row per embedded resource), making sure the entries for *Main Resource Name* and *Main Resource Type* match the label and type previously used when ETLing the main resource. If there are different sub-types for different embedded resources (for example, for Phenotypes), those can be specified in the type field here.
    - Run the ETL command with the embedded resource file as input. Note that the `-p` and `-eid` parameters will be ignored if they are present.

Examples

This Google Drive folder contains a few annotated examples of SWIFT templates with data.

Running the actual ETL

- Determine the type of ETL command that you need to use (create all new or replace/create). ETL commands are described in the SWIFT toolkit guide.
- The ETL process may need to run for a long time (depending on the number of rows in your input, the number of properties filled per row and the network speed) so plan accordingly - e.g. leave it running over night.

> It is good practice to run a test ETL with a subset of rows to detect possible errors in the template data. Run it against a staging server if you have access to one or against the eagle-i training node. If neither of these options is desirable, use your live eagle-i node but ETL the data in the DRAFT state and de-ETL it afterwards.

- We recommend that the ETL process be run using dedicated credentials (e.g. create an "automated curator" user in your eagle-i repository), such that it is later possible to easily isolate resources that were bulk loaded. **ETL needs a Level 4 role.**
- ETL first the secondary resource type files (i.e. linked resources), then the primary resource type file.

> We find it useful to name the data subdirectories according to the order in which they need to be ETLd, e.g. for iPS Cells:
> ```
> 1-human-subject
> 2-diagnosis
> 3-biological-specimen
> 4-primary-cell-line
> 5-induced-pluripotent-stem-cell-line
> ```

- We recommend to ETL resources into the CURATION state, verify the resources were ETLd correctly, and then publish them using the bulk workflow command. The PUBLISH state may be directly used if you are satisfied with your testing.

Post ETL tasks

- Examine the logs generated by the ETL command for possible errors. A log file will be generated per ETLd file, and the row where the error occurred will be indicated. You may need to re-ETL these rows.
- If an ontology term is not resolved during the ETL process, a triple will be added with the predicate `http://eagle-i.org/ont/datatools/1.0/temp_term_not_found` and the literal value found in the data file.
    - issue the following SPARQL query against your repository to find these instances and correct them via the SWEET:

```
select * where {?s <http://eagle-i.org/ont/datatools/1.0/temp_term_not_found> ?o}
```