Sample Vocabulary Registry API workflows

« Vocabulary Registry API methods Resource IRI resolution service »

This page presents some sample workflows that use the Registry API.

Find all vocabularies owned by a particular owner

Let's say you want to find all vocabularies published by a particular owner. You must know the "short name" of the owner. For example, that could be "ARDC", "AODN", or "GA". If you don't know the possible short names, there's only one way to proceed: you'll need to use the getVocabularies method to get a list of all vocabularies. Each vocabulary in the list includes owner information, and you'll need to have a look through it to find the short name of the owner you want. (You can't use the search method for this purpose, as the search results don't include owner metadata.)

However, if you already know the owner's short name (e.g., you are looking for your own records!), you have two main options:

- Use the getVocabularies method, and do your own filtering of the response based on the owner you want.
- Use the search method. Specify the owner as part of the search terms. For example, to find all vocabularies with owner = "ARDC", you could use this as the value of the filtersJson property:
    - `{"pp":-1,"q":"owner:ARDC"}`

Add a new current version to an existing vocabulary that imports data from PoolParty

For publishing a vocabulary that imports vocabulary data from PoolParty, it is recommended to create the vocabulary in the first place using the Portal. That way, the metadata contained in the PoolParty project will be used to pre-populate the top-level metadata fields and to create related entities not already present in the Registry. (If you want to do this yourself using the API, you'd need the getPoolPartyProjects and getPoolPartyProjectMetadata methods; but note that we don't currently provide user support for these methods.)

However, once the vocabulary exists, it is straightforward to add a new version based on the content contained in the PoolParty project.

1. If you don't already have the vocabulary ID, use the search method or the getVocabularies method to find it.
2. Use getVocabularyByIdEdit to get the full metadata of the existing vocabulary, including all of its existing versions.
3. Using the metadata returned in step 2, iterate over the existing versions: if any of them has status `current`, change that status value to `superseded`. (This is another example of something that the Portal editor does for you.)
4. Insert a new version element, specifying no `id` attribute for the version (which indicates creation), and specifying `status="current" do-poolparty-harvest="true"`. Also specify `do-import="true"` and `do-publish="true"` as required. (Of course, you must also satisfy the additional validation constraints, which means including at least a title and a release date.)
5. Send the modified metadata to updateVocabulary. The vocabulary data will be fetched from PoolParty.

Add a new current version to an existing vocabulary, where the vocabulary data is to be uploaded

1. If you don't already have the vocabulary ID, use the search method or the getVocabularies method to find it.
2. Use getVocabularyByIdEdit to get the full metadata of the existing vocabulary, including all of its existing versions.
3. Use createUpload to upload the vocabulary data for the new version. Take note of the upload ID returned by the API method.
4. Using the metadata returned in step 2, iterate over the existing versions: if any of them has status `current`, change that status value to `superseded`.
5. Insert a new version element, specifying no `id` attribute for the version (which indicates creation), and specifying `status="current"`. Also specify `do-import="true"` and `do-publish="true"` as required. (Of course, you must also satisfy the additional validation constraints, which means including at least a title and a release date.)
6. Within the new version element, insert an access-point element. It should have no `id` attribute, but should have `source="user" discriminator="file"`. Within the access-point element, insert an ap-file element, specifying the upload ID returned in step 2 as the value of the `upload-id` attribute. (Do not specify values for the `format` or `url` attributes; the Registry will determine them for you.)
7. Send the modified metadata to updateVocabulary.

« Vocabulary Registry API methods Resource IRI resolution service »