Getting started with the Vocabulary Registry API

« Vocabulary Registry API Vocabulary Registry model entities »

This page provides a high-level introduction to the Vocabulary Registry API. The first main section outlines guiding principles; the second main section gives examples of how those guiding principles are worked out in practice.

Principles of the API

This section lays out the guiding principles of the API and its implementation. More precise details are given further down this page in the section The details.

Built with HTTPS

The Registry API is web-based; its methods are accessed exclusively using HTTPS.

The SSL certificates for vocabs.ardc.edu.au and demo.vocabs.ardc.edu.au are issued by Let's Encrypt. This means that your client must be able to accept Let's Encrypt certificates. In general, this should not be a problem, but if you are using a very old platform, you may experience a problem related to the root certificate. Please check the Let's Encrypt Certificate Compatibility page for more details. As noted at the DST Root CA X3 Expiration Notice page, if you use OpenSSL, you must use version 1.1.0 or later.

CORS-enabled

The OPTION HTTP method is supported, and all the necessary CORS headers are provided, so you can access the API from JavaScript in your browser.

Built with OpenAPI (a.k.a. Swagger)

The Registry API has been constructed from the bottom up using version 2 of the OpenAPI specification and toolset, also known as Swagger. In practice, this means:

- a description of the Registry API is available in the usual OpenAPI machine-readable form (`swagger.json`);
- that description contains documentation of the API methods, their parameters, and their return types;
- that description can be used to access the API using the Swagger UI system, or similar front-ends;
- that description can be used to generate source code for API client libraries in a wide variety of programming languages.

Built with XML Schema (but we also speak JSON)

The Registry API methods consume and produce structured data. The structures used by the API methods are described using XML Schema. In practice, this means:

- a description of the API data structures is accessible in XML Schema format;
- if desired, the schema can be used to validate data before sending it to API methods; conversely, API clients can rely on the structure of data returned by API methods.

The technology that provides marshalling and unmarshalling of data accepts both XML and JSON formats. That means:

- an API client may choose to send request data to the API in either XML or JSON format, and may choose to receive response data in either format, and the request and response formats are even allowed to be different;
- if sending or receiving JSON, the structure of such JSON data is consistent with the equivalent XML structure;
- the Swagger UI system supports working with both XML and JSON, as both request and response data.

If you are accessing the API using the Swagger UI system, or more directly using a tool such as Postman, you will be able to work with your choice of XML or JSON.

If you are using an API client library that has been generated from the OpenAPI description, you usually don't have to worry about the serialisation format – the client library takes care of that for you. For example, if you use a client library generated in Java, the generated code includes model classes that represent vocabulary metadata, and the client library does the work of marshalling and unmarshalling of instances of those classes for transport to/from the API.

Built with authentication and authorisation

Some API methods are public, meaning that they can always be invoked by anyone, without the need to provide authentication, or otherwise to "log in". In general, methods that perform queries are public.

Access to some API methods requires authentication. In general, these are methods that perform modifications (creation/update/deletion) of Registry data.

In addition, some API methods apply additional authorisation checks based on the organisations to which the API user belongs.

Inspired by REST principles

It is *not* a design goal of the API to make it conform precisely to the REST principles. (Conformity to those principles turns out to be very complex, so we prefer not to claim it.) However, there are some "REST-like" characteristics of the API methods. In particular, the Registry has a notion of "resource", and:

- a resource is identified by an API URL that includes the resource type and a resource ID;
- instances of some resource types can be created, queried, updated, or deleted by applying the corresponding HTTP method (POST/GET/PUT/DELETE) to the resource's URL.

The details

OpenAPI

These pages provide background information about OpenAPI v2:

- The OpenAPI Initiative: https://www.openapis.org/
- Specification: https://github.com/OAI/OpenAPI-Specification/blob/master/versions/2.0.md
- Swagger UI: https://github.com/swagger-api/swagger-ui
- Code generation of API client libraries (swagger-codegen): https://github.com/swagger-api/swagger-codegen
    - For more details about how to get started with swagger-codegen, including examples, see the section "Generated API client libraries" at Accessing the Vocabulary Registry API.

The Registry API description is available in OpenAPI v2 format here: https://vocabs.ardc.edu.au/registry/swagger.json. (For the demo environment, the URL is https://demo.vocabs.ardc.edu.au/registry/swagger.json.)

Research Vocabularies Australia hosts a customised instance of Swagger UI, linked directly to the Registry API: http://vocabs.ardc.edu.au/registry/swagger-ui/. (For the demo environment, the URL is http://demo.vocabs.ardc.edu.au/registry/swagger-ui/.)

XML Schema

See Vocabulary Registry data for links to the XML Schema files.

Authentication

As noted above, public methods of the API do not require authentication. If you only need to use the API's public methods, you may skip this section.

Portal users log in using the Portal's "My Vocabs Login" function. This creates an authentication token which is then used as an "API key" to authenticate to the Registry API. The use of this mechanism for systems other than the Portal is not currently supported, and so will not be discussed further here.

At present, the publicly-supported means of authentication to the Registry is by providing the username and password of an ARDC "built-in user" through the HTTP "Basic" authentication scheme, i.e., using the "Authorization" HTTP header. (See, e.g., https://en.wikipedia.org/wiki/Basic_access_authentication for more information about the "Basic" authentication scheme.)

Such "built-in users" are managed in the ARDC roles database. If you would like to use the authenticated methods of the API directly yourself, you will need to acquire a "built-in user" account from ARDC, and then provide its username and password using the "Basic" authentication scheme. To apply for an account, please contact services@ardc.edu.au.

If you use Swagger UI, you authenticate by clicking one of the buttons that says "Authorize" or that contains a padlock icon, then entering the username and password into the "Basic authorization" section and clicking the "Authorize" button.

If you use Postman or a similar REST API client, use that tool's normal means of specifying the username and password for "Basic" authentication.

If you are using an API client library, it will have its own means of specifying the username and password; you won't need to construct the "Authorization" HTTP header yourself.

Example: Java API client library, using Basic authentication

if you are using the generated Java API client library, you can do something like this:

```
ApiClient defaultClient = Configuration.getDefaultApiClient();

defaultClient.setUsername("myusername");
defaultClient.setPassword("mypassword");

ResourcesApi apiInstance = new ResourcesApi();
// And now invoke the methods of apiInstance.
```

Miscellaneous notes

- Why not version 3 of the OpenAPI specification?
  - During the development of the Registry, there was not yet a stable release of all the components of the Swagger toolchain needed to support working with OpenAPI version 3.

« Vocabulary Registry API Vocabulary Registry model entities »