Vocabulary Registry model entities

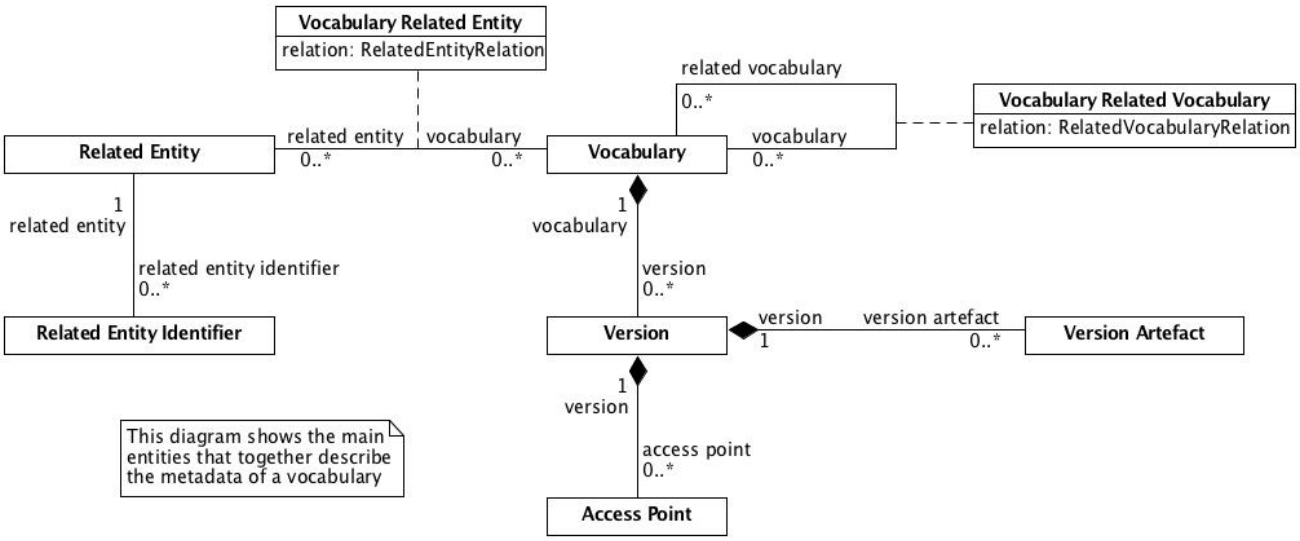
« Getting started with the Vocabulary Registry API Accessing the Vocabulary Registry API »

In order to use the Registry API, you need to have at least a basic familiarity with the underlying vocabulary metadata model. This page provides an introduction to the main vocabulary model entities of the Registry, and how entity instances are related.

Overview

The following UML class diagram shows the main entities in the vocabulary metadata model. (As this is a high-level description, almost all attributes, and all methods, have been omitted from this class diagram.) Don't worry if you don't understand this diagram now; its meaning should become clearer as you read through the rest of this page.

The main entity class is **Vocabulary**. Each instance of the **Vocabulary** class represents the top-level metadata of one vocabulary. All other metadata for one vocabulary is represented in instances of the classes related to the **Vocabulary** class.



In addition, there is an entity class **Upload** that represents uploaded vocabulary content.

Resources and their resource IDs

The Registry API presents instances of some of the entity classes as *resources*. Each such resource has a *resource ID*. For example, the API might present one instance of the **Vocabulary** entity class as the resource `vocabularies/3`, where 3 is the resource ID of the **Vocabulary** instance.

Most of the entity classes shown in the diagram have an attribute that is a resource ID. Resource IDs play an important part in the data sent back and forth between the Registry API and its clients. This is explained in more detail at the page Vocabulary Registry data. Of the entity classes shown in the diagram, these are the ones whose instances have resource IDs:

- **Vocabulary**
- **Version**
- **Access Point**
- **Version Artefact**
- **Related Entity**
- **Related Entity Identifier**
- Instances of the **Upload** class also have an ID which is treated by API methods in a similar way to a resource ID.

A resource ID is an integer value, greater than or equal to 1. Resource ID values are assigned independently to individual instances of each entity class, and there is no relation between ID values of different entity classes. For example, there can be both a **Vocabulary** instance with resource ID 3 and a **Versi on** instance with resource ID 3, but the **Version** instance with resource ID 3 may or may not be associated with the **Vocabulary** instance with resource ID 3.

Vocabulary

An instance of the **Vocabulary** class represents the top level of the metadata of a vocabulary. It contains the vocabulary's title, description, language(s), subject(s), etc.

An instance of the **Vocabulary** class has a *status*: either published, deprecated, or draft. At any one time, a particular vocabulary may exist as just one instance (with any of the three status values), or as two instances with different status values: either published and draft, or deprecated and draft. (In particular, a vocabulary may not have both a published and deprecated instance at the same time.)

Version

An instance of the **Version** class represents one version of a vocabulary. (You can think of a version as a particular "release" of the vocabulary's data.) The **Version** instance contains the version's title, version notes, etc.

In the class diagram, there is a line between the **Vocabulary** and **Version** classes, annotated with "1" at the **Vocabulary** end, and "0..*" at the **Version** end. These values are *multiplicities*. In this case, every **Vocabulary** instance is related to 0 or more **Version** instances. Conversely, any **Version** instance is related to exactly 1 **Vocabulary** instance.

In addition, the **Vocabulary** end of the line is annotated with a filled-in diamond. This makes explicit the link between the *lifetimes* of **Version** instances and their corresponding **Vocabulary** instance. A **Version** instance can not exist independently; it always belongs to one, and only one **Vocabulary** instance. if the **Vocabulary** instance is deleted, so are all of its associated **Version** instances. (The UML terminology for this concept and notation is *compo sition*.)

Access Point

An **Access Point** is a means of "access" to the vocabulary data belonging to a **Version**. For example, an access point may be an external web page, or a direct download link. For each **Access Point** belonging to a **Version**, there will be at least one link associated with that **Version** on the vocabulary's Portal view page. (In the case of an **Access Point** of type `sesameDownload`, there will be one link for each supported RDF format.)

Every **Access Point** has an attribute `source`, which indicates its origin:

- For some types of access point, the Portal or API user may directly request the creation of an access point by specifying it when adding/updating a vocabulary. These have `source` = `user`.
- For some types of access point, the system is responsible for creation of some instances. These have `source` = `system`.

Please note that the multiplicity "0..*" shown on the line between **Version** and **Access Point** reflects the fact that there are circumstances in which, at least conceptually, a Version instance may be *temporarily* without any access points. (This can happen during workflow processing.) However, in general, it is a rule of the Registry that every **Version** have at least one **Access Point**.

Related Entity

A **Related Entity** is a party, service, or external vocabulary (i.e., a vocabulary *not* hosted in RVA), that may have a relationship with a vocabulary. For example, a person or organisation who publishes a vocabulary would be represented as a **Related Entity** of type `party`.

A **Related Entity** instance contains the related entity's title, e-mail address, etc.

In the Registry, **Related Entity** instances are created, updated, and deleted independently from **Vocabulary** instances. To be specific: there is no Registry API method to create, update, or delete a **Vocabulary** and a **Related Entity** at the same time. The Portal CMS manages this for users by making the appropriate API calls to the Registry. (The Portal does not have a separate facility to add/edit/delete related entities apart from the related entity modal dialog provided on the vocabulary CMS page.) If you are using the API in your own program, this is something you may need to pay attention to.

Related Entity Identifier

A **Related Entity** may have identifiers associated with it. A **Related Entity Identifier** contains the identifier's type and value. Supported types include DOI, ORCID, URI, etc. Identifier values are constrained in a type-specific way; for example, if the type is DOI, the value must begin with "`10.`".

Vocabulary Related Entity

Vocabularies are related to **Related Entities** via instances of an "association class", shown in the diagram as **Vocabulary Related Entity**. The basis of the association between any **Vocabulary** instance and an associated **Related Entity** instance is the `relation`, whose values are instances of an enumerated type **Related Entity Relation**.

Each **Related Entity** may be associated with any number of vocabularies. Conversely, it is a rule of RVA that every vocabulary must be associated with at least one **Related Entity** that is a party, where there is the relationship `publishedBy` between the vocabulary and the **Related Entity**.

Vocabulary Related Vocabulary

The RVA entity model distinguishes between relationships between a **Vocabulary** and some other vocabulary not hosted in RVA, and with other **Vocabula ry** instances that *are* hosted in RVA. In the Portal CMS, the two different types are shown as "External Vocabulary" and "Internal Vocabulary".

The **Vocabulary Related Vocabulary** class is an association class; each instance relates one **Vocabulary** instance to another. The basis of the association between any **Vocabulary** instance and another **Vocabulary** instance is the `relation`, whose values are instances of an enumerated type **Related Vocabulary Relation**.

This is a *directed* relationship: **Vocabulary** A may be related to **Vocabulary** B, but this does not automatically mean that **Vocabulary** B is related to **Vocabulary** A.

Version Artefact

A **Version Artefact** represents data associated with a version that is managed internally by the Registry. For example, if a **Version** has vocabulary data in SKOS format, the preferred labels are extracted and stored in a file, and the content of that file is used during the subsequent vocabulary indexing process, so that you can then search for the **Vocabulary**'s concepts in the Portal. That file of preferred labels is managed by the Registry as a **Version Artefact**.

In general, you don't need to pay attention to **Version Artefacts**, and most Registry API methods that accept or produce **Vocabulary** metadata do not deal with them at all.

Upload

An **Upload** represents vocabulary data that has been directly uploaded to the Registry. An **Upload** instance may be referred to by an **Access Point** of type `file`.

« Getting started with the Vocabulary Registry API Accessing the Vocabulary Registry API »