REST API v1

This document applies to COmanage Registry version 4.x and earlier.

- Overview
- Authentication
  - Adding a New API User (Registry v3.3.0 and later)
  - Adding a New API User (Earlier Versions)
- Object Formats
  - URL
  - JSON
  - XML
  - VOOT
  - Request and Response Formats
- Timezones
- Character Sets
- Foreign Keys
- API Reference
- OpenAPI (Swagger)
- Sample Clients

Overview

The Registry Core REST API generally provides table level access to the Registry Data Model. The Core REST API is described in this document, and is designed for accessing and managing data objects, not the Registry configuration. Note that some Core APIs are provided by Plugins, some of which may need to be enabled in order to be used.

In addition, higher level APIs are provided, usually by Plugins. These APIs provide the ability to execute function oriented actions. For more information, see the API specific documentation:

- MFA Status, via MEEM Enroller
- SOR-Registry Write API, via API Source
- VOOT API (deprecated)

Authentication

Authentication is via a proxy or delegated model, where the REST client is treated as an administrative user by the Registry. The client, where appropriate, indicates which target subject it wishes to act on behalf of.

The REST client is authenticated via a simple user/password pair transmitted over HTTPS as part of a basic auth transaction. More sophisticated authentication mechanisms, such as delegated SAML assertions, may be supported in the future.

Note: POST may work for edit due to default CakePHP functionality but is not supported.

Adding a New API User (Registry v3.3.0 and later)

As of Registry v3.3.0, there are three types of API Users:

- **Platform API Users**: API Users created within the COmanage CO are given full access to the API, across all COs.
- **Privileged CO API Users**: API Users created within any other CO may be designated as *Privileged*, in which case they will have full access to the API within their CO.
- **Unprivileged CO API Users**: API Users not designated as Privileged will not have any access to the API by default, but may be granted specific access where supported, for example within a specific plugin.

API Users can be managed by a CO Administrator via *CO >> Configuration >> API Users*. Platform API Users are created the same way, via the COmanage CO.

API Users must have usernames prefixed with the name of the CO, followed by a dot. For example: `MyCO.apiuser`

Self-selected passwords are no longer supported for API Users. Instead, after the API User is created an API Key may be randomly generated. The API Key will be displayed once after generation, but is then hashed for internal storage and is unrecoverable. A new API Key can be generated if needed.

It is also possible to attach validity dates to API Users, as well as to constrain access to specific IP Addresses (via regular expressions). Note there is no current reporting or notification mechanism to indicate the approach of an API User's expiration.

Adding a New API User (Earlier Versions)

Prior to Registry v3.3.0, all API Users have full access to all Registry data across all COs.

Platform Administrators may add and manage API Users via *Platform >> API Users*.

Note that the API User Name must not conflict with any login identifier for any valid user on the platform. This will be enforced when an API User Name is added or edited, but not currently at any other point. (ie: It is possible for a subsequently added person to have a login identifier that conflicts with an API User Name.) (CO-104)

It may make sense to, by policy, only allow login identifiers in eppn format (with an `@`) and to only allow API User Names not in that format (without an `@`).

Object Formats

The REST API supports different formats for representing data object passed. Each format may convey the following special variables:

- **Object Type**: The type of object represented in the request, as defined for each data type.
- **Object Version**: The version of the object represented in the request, as defined for each data type.

URL

For methods such as GET that pass arguments as part of the URL, arguments are positional as defined for each data type.

JSON

This format is supported for requests and responses.

Requests with a JSON body must be sent with a `Content-Type` header of `application/json`.

XML

This format is supported for requests and responses.

The XML format is deprecated as of Registry v3.1.0, and will be removed in Registry v5.0.0 (CO-1555).

VOOT

This format is experimental. See VOOT API for more information.

Request and Response Formats

Changelog

In addition to the attributes defined in the Response formats for each Model, Models enabled for Changelog Behavior will return Changelog metadata as well (`deleted`, `revision`, parent ID, etc).

Note that a request for a deleted, changelog enabled object will return a record (ie: a 200 response, not a 404 response). Examine the `deleted` attribute to verify the status of the object. This behavior is subject to change in a future release (CO-1557).

Attribute Enumerations

If Attribute Enumerations are defined for an attribute, permitted values for that attribute are constrained to the enumerated values. Permitted values may be determined and set via the AttributeEnumeration API.

Timezones

All times processed (inbound and outbound) via the REST API are in UTC. For more information on timezones, see Understanding Registry Timezones.

Character Sets

In general, COmanage supports Unicode, and that includes the REST API. In general, as long as every component of the installation is set up for Unicode (PHP, Apache, the database server) everything should just work. If specifically using a `Content-Type` of `application/json; charset=utf-8`, it may be necessary to use JSON-style `\u####` encoding of non-ASCII characters.

Foreign Keys

As of Registry v3.3.3, updating of foreign keys (attributes of the form `foo_id`) over the REST API is generally restricted. For more information, see Unfreezing Foreign Keys.

API Reference

API	API Version	Available Since Registry	Notes
Address	1.0	v0.1	
AdHocAttribute	1.0	v3.3.0	XML format not supported
ApplicationPreference	1.0	v4.0.0	Limited purpose API in support of Registry user interface
AttributeEnumeration	1.0	v2.0.0	Removed in Registry v4.0.0
Cluster	1.0	v3.3.0	
CO	1.0	v0.1	
COU	1.0	v0.2	
CoDepartment	1.0	v3.1.0	
CoEmailList	1.0	v3.1.0	
CoEnrollmentAttribute	1.0	v0.6	
CoExtendedAttribute	1.0	v0.2	
CoExtendedType	1.0	v0.6	
CoGroup	1.0	v0.1	
CoGroupMember	1.0	v0.1	
CoInvite	1.0	v0.1	
CoNavigationLink	1.0	v0.8.3	
CoNsfDemographics	1.0	v0.4	
CoOrgIdentityLink	1.0	v0.2	
CoPerson	1.0	v0.1	
CoPersonRole	1.0	v0.2	
CoPetition	1.0		
CoProvisioningTarget	1.0		
CoService	1.0	v3.1.0	
CoTAndCAgreement	1.0	v2.0.0	
CoTermsAndConditions	1.0	v2.0.0	
EmailAddress	1.0	v0.1	
HistoryRecord	1.0	v0.7	
Identifier	1.0	v0.1	
IdentityDocument	1.0	v4.0.0	XML format not supported
Name	1.0	v0.8.3	
NavigationLink	1.0	v0.8.3	
Organization	1.0	v4.0.0	XML format not supported The original schema for Registry v0.1, never implemented, is here
OrgIdentity	1.0	v0.2	
Password	1.0	v3.3.0	Implemented by Password Authenticator Plugin, experimental
SshKey	1.0	v3.3.0	Implemented by SSH Key Authenticator Plugin, experimental
TelephoneNumber	1.0	v0.1	
Unix Cluster	1.0	v3.3.0	Implemented by Unix Cluster Plugin, experimental
Unix Cluster Account	1.0	v3.3.0	Implemented by Unix Cluster Plugin, experimental
Unix Cluster Group	1.0	v3.3.0	Implemented by Unix Cluster Plugin, experimental

Url	1.0	v3.1.0	

OpenAPI (Swagger)

An initial, incomplete, and still evolving OpenAPI 3.0.x YAML description of the API is available in the develop branch.

You may use the Swagger container image to render the OpenAPI YAML:

```
git clone https://github.com/Internet2/comanage-registry.git
pushd comanage-registry
git checkout develop
pushd app/Config/Schema

docker pull swaggerapi/swagger-ui
docker run \
    -d \
    --name swagger-ui \
    -p 80:8080 \
    -e SWAGGER_JSON=/opt/restapiv1.yaml \
    -v ./restapiv1.yaml:/opt/restapiv1.yaml \
    swaggerapi/swagger-ui
```

Then browse to http://localhost to see the rendered API.

You may also use the OpenAPI 3.0.x YAML description with the OpenAPI generator to generate client bindings for various programming languages. See REST API Examples for examples.

Sample Clients

See REST API Examples.