REST API v2

This document applies to COmanage Registry version 5.0.0 and later. For earlier versions, see REST API v1.

About the REST API

The Registry Core REST API, documented here, is designed to allow fine grained resource access to the Registry database. In other words, the REST API interfaces align more or less with the objects stored within, and not to "higher level" abstractions. Note that some Core APIs are provided by Plugins, some of which may need to be enabled in order to be used. The Core REST API is not optimized for bulk transactions. A better approach for bulk transactions is to use External Identity Sources.

The API is available at https://server/registry/api/v2/.

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:

- API Source: Provides an External Identity Source interface to load data from upstream data sources

Authentication

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.

Passwords are generated as *API Keys* associated with each API User, as described below.

Adding a New API User

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 can also be managed via the API User API.

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

Self-selected passwords are not 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.

Object Formats

v2 of the API only supports JSON for request and response payloads. XML is no longer supported. The VOOT API is no longer supported.

The requested object format is no longer specified as an extension on the URL. Instead, all requests must include an `Accept` header of `application/json`.

Timezones and Timestamps

All times processed (inbound and outbound) via the REST API are in UTC (AR-GMR-4). For more information on timezones, see Registry Timezones.

Timestamps are returned in `YYYY-MM-DDTHH:MM:SS` format (with no timezone indicator).

Foreign Keys

In general, foreign keys cannot be updated using the REST API to move objects across COs. For example, `CoPerson:co_id` cannot be updated after it is set.

Frozen Attributes

The REST API can be used to mark an attribute as frozen (by sending `frozen: true`), but once an attribute is frozen it can no longer be directly updated (and the REST API will return a `401 Unauthorized`). There is not currently a mechanism to unfreeze attributes via the REST API, this must be done via the frontend.

Request Formats

Requests are provided as an object named for the appropriate model, and an object (for Update requests) or a list of objects (for Add requests) holding model specific attributes (excluding metadata and `id`).

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

Request Headers

As described above, any request that provides a JSON body (typically `PUT` or `POST`) must include a `Content-Type` header. Any request that receives a JSON body (including `GET`) must include an `Accept` header. If either is omitted when required, SecurityComponent errors will result and the request will be blackholed.

```
Content-Type: application/json
Accept: application/json
```

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.

Response Formats

Response Metadata

Each response includes an object called `responseMeta` with metadata about the response. Response metadata attributes include:

- `resource`: The name of the resource (or object) being returned
- `version`: API version, currently "2"

In addition, Response Metadata may include pagination information, documented below.

Record Metadata

Individual records include a `meta` object with metadata about the record. Record metadata attributes include:

- `actor_identifier`: The login identifier associated with the last record modification*
- `created`: The timestamp of the record creation
- `deleted`: If `true`, the record is deleted*
- `modified`: The timestamp of when the record was last modified
- `revision`: The revision number of this record*
- *`attribute`*`_id`: If set, this is an older revision of the specified record*

*See Changelog Behavior for more information.

Pagination

Pagination is generally available on "index" listings of objects, ie: requests that can return more than one object in a list. Pagination is controlled by appending the following parameters to the query string:

- `direction`: Controls the direction of the sort of objects, `asc` (ascending) or `desc` (descending)
- `limit`: The maximum number of results to return (per page)
- `sort`: The field to sort the results on
- `page`: Number of page to return results for, with 1 being the first page

The following attributes will be returned in the Response Metadata:

- `currentPage`: The current page of results, which should correspond to the `page` parameter
- `itemsPerPage`: The number of objects in this page, which should correspond to the `limit` parameter
- `pageCount`: The total number of pages
- `startIndex`: The index of the first result (out of totalResults) in the current page, starting from 1 (the first result on the first page)
- `totalResults`: The total number of results

It is possible to guarantee a full pagination over a large dataset that might receive new records during the pagination process by using Keyset Pagination. In short, sort on the *id* field in *asc* (ascending) direction. New records will be appended to the set as each new page is returned (as indicated by `totalResults`).

Standard API Operations (CRUD)

Most APIs implement a standard set of CRUD operations, which are documented here. See the documentation for each API (under *API Reference*, below) for the details of which attributes are supported for each API, as well supplemental API calls or deviations from the standard CRUD set.

Add

Add one or more new objects.

> ⚠ When more than one object is provided, and the object type has a parent, all objects must have the same parent value. For example, if adding two COUs in one operation, both COUs must belong to the same CO.

Request Format

Request	`POST /api/v2/`***`objects`***
Request Body	List of objects
Paginated?	No

Response Format

HTTP Status	Response Body	Description
`200 OK`	Results Response	Add requests generate a `results` object with an array of objects correlating to the request objects. For example, if two objects were added, there will be two results in the array, in the order of the original request. Each result is an object `id` on success, or `error` on failure.
`400 Bad Request`	Error Response	Attribute validation failed

`401 Unauthorized`		Authentication required
`500 Internal Server Error`	Error Response	Server error

Example: Add Two New COs

```
POST /registry/api/v2/cos
{
  "Cos": [
    {
      "name": "REST Test",
      "description": "Test REST API v2",
      "status": "Q"
    },
    {
      "name": "REST Test 2",
      "description": "Second Test REST API v2",
      "status": "A"
    }
  ]
}

200 OK
{
  "results": [
    {
      "error": "Entity save failure (status: \"content\")."
    },
    {
      "id": 142
    }
  ]
}
```

Delete

Remove an object. In most cases, this will execute a "soft" delete, meaning the record is logically deleted but an archived copy remains in the database. See Changelog Behavior for more information.

Request Format

Request	`DELETE /api/v2/`***`objects`***`/`***`id`***
Request Body	None
Paginated?	No

Response Format

HTTP Status	Response Body	Description
`200 OK`		Object deleted
`400 Bad Request`	Error Response	*`id`* not provided or not found
`401 Unauthorized`		Authentication required
`500 Internal Server Error`	Error Response	Server error

Edit

Edit an existing object. Note the old values for the object are archived, see Changelog Behavior for more information.

Request Format

Request	`PUT /api/v2/`***`objects`***`/`***`id`***
Request Body	Object
Paginated?	No

Response Format

HTTP Status	Response Body	Description
`200 OK`		Update successful
`400 Bad Request`	Error Response	Attribute validation failed or *id* not found
`401 Unauthorized`		Authentication required
`500 Internal Server Error`	Error Response	Server error

Example: Update CO

```
PUT /registry/api/v2/cos/142
{
  "Cos": {
    "name": "REST Test II",
    "description": "Second Test REST API v2",
    "status": "A"
  }
}

200 OK
```

View (one)

Retrieve an existing object.

Request Format

Request	`GET /api/v2/`***`objects`***`/`***`id`***
Request Body	None
Paginated?	No

Response Format

HTTP Status	Response Body	Description
`200 OK`	An array of a single CO object	CO returned
`401 Unauthorized`		Authentication required
`404 Not Found`		*`id`* not found
`500 Internal Server Error`	Error Response	Unknown error

The response format is the same as for *View (all)* below, but without pagination metadata.

View (all)

Retrieve all existing objects. This request may be modified by API-specific parameters to filter available results.

Request Format

Request	`GET /api/v2/`***`objects`***
Request Body	None
Paginated?	Yes

Response Format

HTTP Status	Response Body	Description
`200 OK`	An array of a single CO object	CO returned
`401 Unauthorized`		Authentication required
`500 Internal Server Error`	Error Response	Unknown error

For requests that generate response bodies (ie: "View" requests), the format is a `responseMeta` object holding Response Metadata, followed by a response object named for the model requested. The response object will be a list of data objects, with a unique `id` field, a set of attributes appropriate for the model, and a `meta` section with Record Metadata.

Example: Obtain COs, Paginated

```
GET /registry/api/v2/cos?page=2&limit=3

200 OK
{
  "responseMeta": {
    "totalResults": "20",
    "startIndex": "4",
    "itemsPerPage": "3",
    "currentPage": 2,
    "pageCount": 7,
    "resource": "Cos",
    "version": "2"
  },
  "Cos": [
    {
      "id": 1,
      "name": "COmanage",
      "description": "COmanage Registry Internal CO",
      "status": "A",
      "meta": {
        "created": "2011-07-28T21:26:09+00:00",
        "modified": "2011-07-28T21:26:09+00:00",
        "revision": null,
        "deleted": null,
        "actor_identifier": null,
        "co_id": null
      }
    },
    {
      "id": 2,
      "name": "TestCO",
      "description": "Test CO",
      "status": "A",
      "meta": {
        "created": "2011-07-28T21:27:52+00:00",
        "modified": "2018-11-27T00:37:47+00:00",
        "revision": null,
        "deleted": null,
        "actor_identifier": null,
        "co_id": null
      }
    },
    {
      "id": 68,
      "name": "Flat CO",
      "description": "CO with no COUs",
      "status": "A",
      "meta": {
        "created": "2018-06-01T02:39:04+00:00",
        "modified": "2018-06-01T02:39:04+00:00",
        "revision": null,
        "deleted": null,
        "actor_identifier": null,
        "co_id": null
      }
    }
  ]
}
```

API Reference

API	Version	Available Since*
AdHocAttribute	2	v3.3.0
Address	2	v0.1

API User	2	v5.0.0
CO	2	v0.1
COU	2	v0.2
CO Settings	2	v5.0.0
EmailAddress	2	v0.1
ExternalIdentity	2	v0.2
ExternalIdentityRole	2	v5.0.0
Group	2	v0.1
GroupMember	2	v0.1
GroupNesting	2	v5.0.0
~~GroupOwner~~	~~2~~	~~v5.0.0~~
HistoryRecord	2	v0.7
Identifier	2	v0.1
Job	2	v5.0.0
JobHistoryRecord	2	v5.0.0
Name	2	v0.8.3
Person	2	v0.1
PersonRole	2	v0.2
Pronoun	2	v5.0.0
Server	2	v5.0.0
TelephoneNumber	2	v0.1
Type	2	v0.6
Url	2	v3.1.0

**For APIs available prior to Registry v5.0.0, column denotes introduction of API v1. All APIs transitioned to API v2 with Registry v5.0.0.*