# REST API

The REST API is a service layer built on top of the PHP API, so most (but not all) operations available in PHP are also available using REST.

## Endpoint

The API is located at `/api` under the root of the Omeka S installation.

## Format

### Responses

HTTP responses will be formatted by default in JSON-LD, a method of transporting Linked Data using JSON.

Since Omeka S version 4.1.0, API clients can request that responses be delivered in other formats by passing a `format` query string parameter. The core supported formats are:

- JSON-LD ( `jsonld` , the default)
- RDF/XML ( `rdfxml` )
- N3 ( `n3` )
- Turtle ( `turtle` )
- N-Triples ( `ntriples` )

### Requests

Payloads for requests should also be JSON-LD, but Omeka S sometimes requires clients to follow a particular structure, even if the same data could be represented by alternate valid JSON-LD structures. The Content-Type of a request with a payload must be `application/json` (or `multipart/form-data` if sending a multipart request). Format specifers (before `+` , i.e., `application/ld+json` , are also allowed).

Parameters are passed to the REST API using the query string, with PHP's convention for passing arrays using square brackets in the parameter names.

## Authentication

The API permits anonymous access to public resources (i.e., reading non-private data). To perform actions or view data that only logged-in users can access, you must authenticate your requests.

Authentication requires every request to include two additional GET parameters: `key_identity` and `key_credential`. An Omeka S user can create API keys in the "API keys" tab of their user edit page.

## API Operations

### Search

```
GET /api/:api_resource
```

Search or list resources, optionally specifying criteria. Parameters to filter, limit, or alter the search results should be passed as GET parameters in the query string. Links to additional pages are given in the `Link` header. The link with `rel="next"` is the next page in the sequence. When appropriate, links with a `rel` value of `prev`, `first`, and `last` are also provided.

As of Omeka S 3.0.0, search responses include an `Omeka-S-Total-Results` header that indicates the total number of results across all pages.

### Read

```
GET /api/:api_resource/:id
```

Get one resource by a known ID. The `id` parameter is mandatory.

### Create

```
POST /api/:api_resource
```

Create a resource. A JSON payload is required.

**Upload files**

For create operations that involve uploading files, multipart requests (Content-Type `multipart/form-data`) are supported. In a multipart request, the normal JSON request body should be specified as a "field" with the name `data`. The `asset` API resource expects the associated file to be named `file`. To upload media (as part of a `media` create operation or `item` create or update operation), files are expected to be named as an "array" (i.e., `file[0]`, `file[1]`). The JSON body for the media or item will specify the index into this array with the `file_index` key.

```
"o:ingester": "upload",
"file_index": 0,
...
```

### Update

```
PUT /api/:api_resource/:id
```

Update a resource. The `id` URL parameter is required. A JSON payload is required. The payload completely updates/replaces the existing resource with the given ID. Therefore, if making changes, it's typically wise to read the current state of the resource, alter the returned JSON, and submit that as the update. Alternatively, you can perform a "partial" update.

### Partial Update (Patch)

```
PATCH /api/:api_resource/:id
```

Update a resource, preserving data in non-specified keys. The `id` URL parameter is required. A JSON payload is required. Note that specifying *any* values for RDF properties (the "values" for items, item sets, and media) will be treated as an "update" to the values generally (meaning that specifying any value for any RDF property will mean the removal of all other values that aren't passed). In other words, the values are treated as if they're one big "key" for the purposes of patching.

### Delete

```
DELETE /api/:api_resource/:id
```

Delete a resource. The `id` URL parameter is required.

## Examples

### List items

URL to get the JSON-LD for all items with a GET request, as a paginated list:

```
/api/items
```

The response is an array of JSON-LD objects, one for each item. One page of the full result set will be returned, with additional pages being available using the `page` parameter:

```
/api/items?page=2
```

Parameters can be added to filter and sort the result set. (See the API Reference for possibilities). For example, to list all items using the resource class with ID 2, ordered by their item ID with the highest IDs first:

```
/api/items?resource_class_id=2&sort_by=id&sort_order=desc
```

To get the same results but further filtered to only include items that contain the string "foo" in the dcterms:title property, using the `property` parameter (with newlines inserted between the parameters for clarity):

```
/api/items
    ?resource_class_id=2
    &property[0][property]=dcterms:title
    &property[0][type]=in
    &property[0][text]=foo
    &sort_by=id
    &sort_order=desc
```