

## API Version 1 Documentation

Overview.....	2
API Base URL.....	2
Authorization.....	2
Making an API Request .....	3
Retrieving JSON Files.....	4
Site Snapshots.....	6
GET Full Site Snapshot .....	6
GET Monthly Update Snapshot .....	7
Filter Snapshots.....	8
PUT Create a Search-Based Filter.....	8
GET Get Results for a Filter .....	11
GET Get a List of Current Filters.....	12
DELETE Delete a Filter .....	14
Filtering Based on Authors.....	14
Ensuring the Best Results .....	15
Bulk Adding Institutional IDs to Authors.....	15
Creating Filters Based on Author ID.....	16
PUT Add an ID to a Specific Author.....	17
GET Get Author Metadata for Specific Author ID.....	18
GET Get All Author Metadata for Your site .....	18
Common Filter Queries.....	19
Metadata for Author Name in a Specific Series.....	19
Metadata for an ORCID ID (previously added to an author via the API).....	20
Metadata for Specific Departmental Series and Year.....	20
Metadata for Most Recent Issue of Journal.....	20
Metadata for ETDs with a Specific Keyword .....	20
Metadata for a Specific ETD Advisor in a Department.....	20
FAQ.....	21

# Overview

The Digital Commons outbound API (version 1) allows you to retrieve metadata for your repository to use in a variety of applications. An introductory guide with common uses is available on our website at [https://www.bepress.com/reference\\_guide\\_dc/digital-commons-api/](https://www.bepress.com/reference_guide_dc/digital-commons-api/).

As a RESTful API, the Digital Commons API uses a base URL and standard HTTP methods such as GET, PUT, and DELETE. Each request requires two special headers (API key and security token) and, in the case of filtered snapshots, a body (the filter parameters).

Version 1 of the API is asynchronous and refreshed monthly with the latest data. You can call the current data from the API as many times as needed. Endpoints include both a full snapshot and an incremental update snapshot, with output in JSON files. All available metadata (including hidden metadata) is returned in API results.

In addition to the site snapshots, you may retrieve metadata for specific content by creating filters that use Solr-based field search criteria. You're also able to assign institutional IDs or identifiers such as ORCID IDs to authors and build filters based on ID criteria.

If you have any questions about the API, please contact Consulting Services:  
[dc-support@cs.bepress.com](mailto:dc-support@cs.bepress.com)

## API Base URL

The base URL of the Digital Commons API is:

```
https://content-out.bepress.com/v1
```

Endpoints listed in this document are appended to this URL when making API requests.

Request URLs must be called along with the headers described under "Authorization," using an HTTP client or your choice of programming language. **Request URLs will not work if pasted into a browser.**

## Authorization

You will need to include an API key and security token as headers with each request. Please contact Consulting Services to obtain your key and security token.

`X-API-Key` is the header name for the API key

`Authorization` is the header name for the security token

The headers follow this pattern:

```
X-Api-Key:API_KEY  
Authorization:SECURITY_TOKEN
```

Replace `API_KEY` with the API key you received and `SECURITY_TOKEN` with your security token. Remember to include the API key and security token in the headers of each request, and not in the request URL.

## Making an API Request

You may use your preferred client or programming language to call the API. This documentation includes instructions for a free client called [Postman](#). Other common clients (also free) include [HTTPie](#) and [Insomnia](#).

API requests follow this pattern, with a method (GET in this case), an endpoint including a site URL, and headers:

```
GET https://content-out.bepress.com/v1/ENDPOINT/YOUR_SITE_URL  
X-Api-Key:API_KEY  
Authorization:SECURITY_TOKEN
```

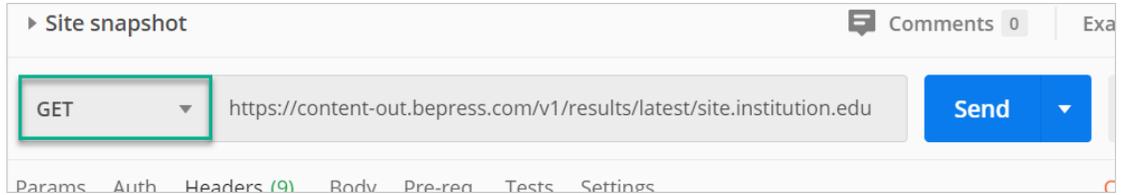
Here are the basic steps for making an API request:

1. Combine the API base URL with the desired endpoint. Endpoints are listed in this documentation along with example request URLs.
2. Replace `YOUR_SITE_URL` in each endpoint with the URL of your Digital Commons site minus "https://", e.g., `your.institution.edu`.
3. Include the `X-Api-Key` and `Authorization` in the request headers, as described above.
4. Enter parameters in the body if creating a search-based filter, as described in the filter instructions.

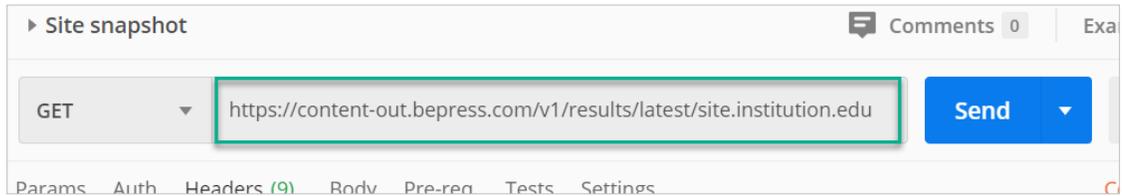
Making requests in Postman:

If using the Postman client, here are some pointers for where to put the various components of an API request.

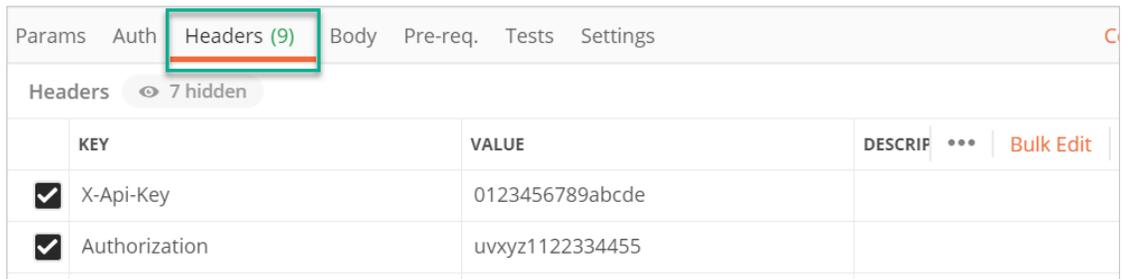
1. Select the method (GET, PUT, or DELETE) for your request using the Postman method dropdown menu. The endpoints listed in this documentation specify the HTTP method for each type of request.



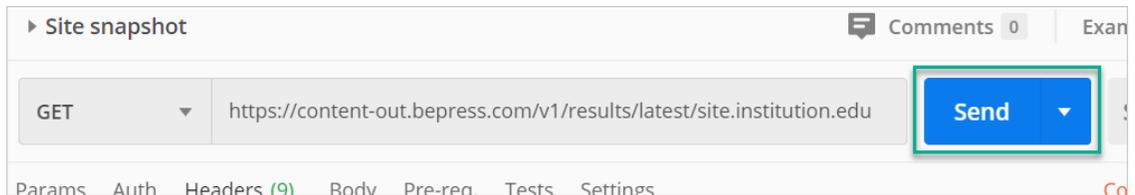
2. Add the URL request to the Postman request window. Example URL requests are provided for the endpoints listed in this document.



3. Click on **Headers** to add the API key, with X-Api-Key as the **Key**, and the actual API key in the **Value** field. In another row within **Headers**, add your security token: Enter Authorization as the **Key** and your security token in the **Value** field.



4. If creating a filter, you'll also add search parameters, which go in the body: click on **Body**; then click the radio button next to **raw** and choose **JSON** from the dropdown to the right. Paste your search parameters.
5. Click the Send button.



## Retrieving JSON Files

When making a GET request for a snapshot of records, the response will include a URL with the location of a file containing the requested data in JSON format. Please note that links returned by the API for access to files expire after 24 hours.





```
QetqCt7qP%2FiUVVGvWwiv7eIsBxi%2F5h7JwwLraEP9JrAgTe3UKIt77iqJJCuzo
e%2B81shBaClriqumDYf7%2BFBjMM%2BNyPUF0t8BZnD40yyPi%2F2W1dxBbW6UR%
2Fi%2BnDbj%2Bvn9BN%2FW9VM0ztfJ7jDZ0fX3Lz9jmn6Zq0ByX0wGKkrI4gYvxzP
EOqazKNW3WxygN2P5cNb4Sd9IigvjwVBwB%2FWYeFm9Xph1cVbnNJXU%2FjZEtBwt
Y4c8voS9b8BV2IfrJRQkOgIJGKBvTD1vvXMGiW9jSxFNp3g3DTGW%2F3f4h%2Fenn
jY7Ez1nWRC8W9B4x6V9%2FFGrpyU8OrgCa8GTL%2FXiYuvUNLRpuQMue%2FWQ9j4H
aCCrEATGY%2Few9e3d05WE4%2FvCe0SXuZh7pgkU%2Bg%3D%3D&Expires=158881
1859"
  ]
}
```

### GET Monthly Update Snapshot

Retrieves metadata (including hidden metadata) for any items that have been added or updated since last month. Like the full snapshot, you may access it as often as you want, but it will be refreshed monthly between the 1<sup>st</sup> and the 5<sup>th</sup> of the month.

#### Endpoint

```
/results/monthly/YOUR_SITE_URL/YYYY-MM
```

#### Example request URL

Example call for March 2020:

```
https://content-out.bepress.com/v1/results/monthly/site.institution.edu/2020-03
```

#### Method

GET

#### Steps

1. Make a GET request with the above endpoint URL. Replace YOUR\_SITE\_URL with the URL of your Digital Commons site minus "https://". Include the API key and security token as headers. See "Making an API Request" for detailed steps.
2. The API will return a URL with the location of a JSON file (the link expires after 24 hours).
3. Download the JSON file that has been generated with your data. See "Retrieving JSON Files" for detailed instructions.
4. Open the JSON file using your preferred program.



- Compound filter snapshots are filtered by more than one parameter, for example by publication date and keyword. An instance where you might want to use a compound filter snapshot is if you need to pull data for all items that were published in 2019 with the keyword “literature.”

## Endpoint

```
/filters/YOUR_SITE_URL/FILTER_ID
```

Replace `FILTER_ID` with a unique filter name of your choice, such as “published-2019”.

## Method

PUT

## Example request URLs

Simple and compound filters follow the same request URL format:

```
https://content-out.bepress.com/v1/filters/site.institution.edu/published-2019
```

## Body parameters

Filters require parameters to tell the API what to return; these will go in the body of the PUT request. The allowed parameters are:

- “op” (operator), which can be
  - “equals” – an exact match for a field with a single value; “equals” is required for numbers such as `author_userid` and for Boolean values
  - “contains” – a match for a substring in a field with a single value (e.g., to search for words or phrases in abstracts or titles)
  - “has-item” – an exact match for a single item in a multiple value field (or one that must be treated as multiple); “has-item” is required for author ID parameters (see “Filtering Based on Authors”)
- “field”, which is the name of the metadata field to be filtered (based on back-end label for chosen metadata field)
- “value”, which is the specific value on which to filter the field

Simple filter body example:

```
{"filters":[{"op":"contains", "field":"publication_date", "value":"2019"}]}
```

Compound filter example:

```
{"filters":[{"op":"contains", "field":"publication_date",  
"value":"2019"}, {"op":"contains", "field":"abstract", "value":"literature"}]}
```

## Steps

1. Make a PUT request with the endpoint URL; body; and header containing the API key and security token. Follow instructions under “Making an API Request.”
2. Results should look like the samples below. If your initial results contain “null” in the “State” and “CompletedOn” fields, you can confirm that the filter has been created by running a GET request using this pattern:

```
GET API_BASE_URL/filters/YOUR_SITE_URL/FILTER_ID  
X-Api-Key:API_KEY  
Authorization:SECURITY_TOKEN
```

## Sample response

The “State” should show “SUCCEEDED” and “CompletedOn” should show a timestamp. Once those values are returned, creation of your filter is complete. If “State” and “CompletedOn” are still ‘Null’, the filter has not completed its work.

Simple filter, completed:

```
{  
  "site": "site.institution.edu",  
  "filters": [  
    {  
      "Body": {  
        "filters": [  
          {  
            "op": "contains",  
            "field": "publication_date",  
            "value": "2019"  
          }  
        ]  
      },  
      "Site_Id": "site:site.institution.edu",  
      "Filter_Id": "published-2019",  
      "State": "SUCCEEDED",  
      "CompletedOn": "Fri, 08 May 2020 00:09:22 GMT"  
    }  
  ]  
}
```

Compound filter, not yet completed:

```
{
  "site": "site.institution.edu",
  "filters": [
    {
      "Body": {
        "filters": [
          {
            "op": "contains",
            "field": "publication_date",
            "value": "2019"
          },
          {
            "op": "contains",
            "field": "abstract",
            "value": "literature"
          }
        ]
      },
      "Site_Id": "site:site.institution.edu",
      "Filter_Id": "lit-filter",
      "State": null,
      "CompletedOn": null
    }
  ]
}
```

## GET Get Results for a Filter

Retrieve search-based results produced by running a filter for a site.

Results will return all metadata (including hidden metadata) for all published items that meet the parameters of the query.

### Endpoint

```
/results/filters/YOUR_SITE_URL/FILTER_ID
```

The filter ID is defined during filter creation (see “Create a Search-Based Filter”).

### Method

GET

## Example request URL

```
https://content-out.bepress.com/v1/results/filters/site.institution.edu/lit-filter
```

## Steps

1. Make a GET request with the above endpoint URL. Replace `YOUR_SITE_URL` with the URL of your Digital Commons site minus "https://". Include the API key and security token as headers. See "Making an API Request" for detailed steps.
2. The API will return a URL with the location of a JSON file (the link expires after 24 hours).
3. Download the JSON file that has been generated with your data. See "Retrieving JSON Files" for detailed instructions.
4. Open the JSON file using your preferred program.

## Sample response

```
{
  "site": "site.institution.edu",
  "results": [
    "https://s3.amazonaws.com/content-out-client-data-
      v0.1production/outbox/sites/site.institution.edu/filters/lit-
      filter/run-1588635105004-part-r-
      00000?AWSAccessKeyId=ASIAYVUS7KB2HQTYX6WI&Signature=sQw3iIaiSWEMK
      ZDuNJXe0WZTu9Q%3D&x-amz-security-
      token=IQoJb3JpZ2luX2VjEGEaCXVzLWVhc3QtMSJIMEYCIQCPZVi7nYxpprPXFfw
      TTR7nE5FY2MH50Z0IiHacInfMgIhAKWNFe0mip5uXqMAhLXtq2mQq0TpMkPaJ6Yl
      terJqZgYKt4BCJr%2F%2F%2F%2F%2F%2F%2F%2F%2F%2FwEQARoMNTk2MjM00TQ4N
      zI0IgwZYpNDuc9%2F1v0ymvYqsgF8D14Zsz0%2BUbd1rbwajy5dAB2o9Zext%2Bv1
      y3kR0jjViiRGoqsR6ZP1%2Fsre%2BisCriQyGxLPgj9mgsdENPzUZ0wFM00MhRFad
      8lpF%2BGx3xRxYGdm4J4COjEqQgVajWFG1Qj%2B2jjav4u9bcaPJfYwYL%2FkNypv
      QetqCt7qP%2FiUVVGvWwiv7eIsBxi%2F5h7JwwLraEP9JrAgTe3UKIt77iqJJCuzo
      e%2B81shBaClriqumDYf7%2BFBjMM%2BNyPUFOt8BZnD40yyPi%2F2W1dxBbW6UR%
      2Fi%2BnDbj%2Bvn9BN%2FW9VM0ztF7jDZOFX3Lz9jmn6Zq0ByX0wGKkrI4gYvxzP
      EOqazKNW3WxygN2P5cNb4Sd9IigvjwVBwB%2FWYeFm9Xph1cVbnNJXU%2FjZEtBwt
      Y4c8voS9b8BV2IfrJRQkOgIJGKBvTD1vvXMGiW9jSxFNp3g3DTGW%2F3f4h%2Fenn
      jY7EZ1nWRC8W9B4x6V9%2FFGrpyU8OrgCa8GTL%2FXiYuvUNLRpuQMue%2FWQ9j4H
      aCCrEATGY%2Few9e3d05WE4%2FvCe0SXuZh7pgkU%2Bg%3D%3D&Expires=158881
      1859"
  ]
}
```

## GET Get a List of Current Filters

Retrieves a list of search-based filters that have been created for your site.

## Endpoint

```
/filters/YOUR_SITE_URL/
```

## Method

GET

## Example request URL

```
https://content-out.bepress.com/v1/filters/site.institution.edu/
```

## Steps

1. Follow steps under “Making an API Request” to call the API.
2. The response will contain all results. There will be no link with a file to download. (If desired, you can save the response in Postman by clicking “Save Response”, then “Save to a file”.)

## Sample response

```
{
  "site": "site.institution.edu",
  "filters": [
    {
      "Body": "{\\\"filters\\\": [{\\\"op\\\": \\\"contains\\\", \\\"field\\\": \\\"publication date\\\", \\\"value\\\": \\\"2019\\\"}, {\\\"op\\\": \\\"contains\\\", \\\"field\\\": \\\"abstract\\\", \\\"value\\\": \\\"literature\\\"}]}\",
      \"Site Id\": \"site:site.institution.edu\",
      \"Filter Id\": \"lit-filter\",
      \"State\": \"SUCCEEDED\",
      \"CompletedOn\": \"Mon, 11 May 2020 21:32:13 GMT\"
    },
    {
      \"Body\": \"{\\\"filters\\\": [{\\\"op\\\": \\\"has-item\\\", \\\"field\\\": \\\"ancestor link\\\", \\\"value\\\": \\\"https://site.institution.edu/series1\\\"}, {\\\"op\\\": \\\"contains\\\", \\\"field\\\": \\\"publication date\\\", \\\"value\\\": \\\"2019\\\"}]}\",
      \"Site Id\": \"site:site.institution.edu\",
      \"Filter Id\": \"series-2019\",
      \"State\": \"SUCCEEDED\",
      \"CompletedOn\": \"Mon, 11 May 2020 22:08:58 GMT\"
    }
  ]
}
```

## DELETE Delete a Filter

For deleting a search-based filter that was previously created for your site.

### Endpoint

```
/filters/YOUR_SITE_URL/FILTER_ID
```

The filter ID is defined during filter creation (see “Create a Search-Based Filter”).

### Method

DELETE

### Example request URL

```
https://content-out.bepress.com/v1/filters/site.institution.edu/lit-filter
```

### Steps

See “Making an API Request” for more detailed steps if needed.

### Sample response

Returns a list of your current filters, allowing you can confirm that the deleted filter no longer appears.

## Filtering Based on Authors

For cases where you would like to use an author as a filter parameter, the API has an additional feature that allows you to attach an institutional (or other) ID to an author. This is intended to help you better disambiguate your authors and improve the results of your call.

This method allows you to query using an identifier that will return all items by the author associated with that identifier, regardless of whether they are using the same name or not.

These identifiers can be anything—institutional faculty IDs, ORCID IDs, Scopus IDs, etc.—and you can add up to five for any given author. This ID will be the filter parameter when calling the API.

Some use cases in which you may want to identify publications in your repository for a specific author include:

- Adding publications to faculty evaluation systems such as Pure, Symplectic, Digital Measures, or Faculty 180.

- Posting lists of publications to faculty websites
- Pulling data to analyze a specific author's output

## Ensuring the Best Results

The cleaner and more accurate your data, the better your results will be. In order to achieve optimal results from filtering by author, we recommend the following steps before adding institutional or other IDs to your authors:

1. Choose a list of institutional authors for whom you'd like to add author IDs. Possible sources might be a faculty directory, library catalog, or a specific departmental list.
2. Use the Author Merge Tool ([instructions here](#)) to identify any authors who have Digital Commons records without email addresses. Merge each of those authors with a version of the same author that has an email address.
3. If none of the author's records has an email address, you can first revise a record on which they are an author to add their institutional email address. This will create a bepress user ID for those authors, which will be required in order to add additional IDs. (It will also ensure that the author receives monthly readership emails.) You can then use the Author Merge Tool to merge those authors under one account.
4. If an author is listed with **more than one email**, you will need to contact your Consultant to merge those accounts for you. Please cc the author or otherwise pass along author consent to merge accounts when making the request.
5. Once you have completed the above steps to achieve cleaner author lists and accounts, contact Consulting Services ([dc-support@cs.bepress.com](mailto:dc-support@cs.bepress.com)) and let your consultant know that you are ready to add IDs to your authors. You will receive a spreadsheet back with all authors in your repository whose accounts use your institution's email address.

## Bulk Adding Institutional IDs to Authors

At this point, you will have completed initial cleanup and received a spreadsheet above. The spreadsheet contains the bepress user ID (author\_userid), first name, last name, email address, and institutional affiliation for authors in your repository that have an account using your institution's email domain.

You may add institutional or other IDs to the spreadsheet following these steps:

1. Add a column to the spreadsheet for each type of ID you would like to add. When naming the column headers, be sure to use letters, digits, and/or underscores (\_). You can add up to five columns, which will allow you to add up to five IDs per author; the IDs can be whatever you choose.
2. Enter each author's ID(s) in the appropriate cells and save the spreadsheet.
3. Email the spreadsheet back to your consultant for processing.

If any authors are missing from the spreadsheet, look them up in the Author Merge Tool to make sure they have your institutional email associated with their account.

- If they don't have an email address associated, add their institutional email to any records in your repository on which they are an author.
- If they are listed under a different email address, contact your consultant to merge them with an account that uses their institutional email address. Please cc the author or otherwise pass along author consent to merge accounts when making the request.
- If there are still authors that should be there that aren't, please include that information when returning the spreadsheet and we will look into it.

Your consultant will contact you when your spreadsheet has been processed and IDs added to your authors. At that point, you can call the API using any of those IDs as filter parameters (see below).

**Note:** The spreadsheet that is used to bulk add IDs is not a batch revise spreadsheet. If you see data in the spreadsheet that is incorrect, for example an email address or affiliation that needs to be changed, please wait until we've processed your author ID additions and then revise the record in your Digital Commons.

## Creating Filters Based on Author ID

When IDs are added to authors, it will add additional field(s) to the author metadata available through the API. The added fields begin with "ext\_", followed by the label you used when adding the author ID. For example, if you added the label "scopus\_id", the new metadata field will be "ext\_scopus\_id"; an "orcid\_id" label will become "ext\_orcid\_id" and so on.

These metadata fields and their accompanying metadata will only be available via the API; they will not be added to your Digital Commons article metadata records.

To create a filter, follow the steps in this documentation for creating either simple filters or compound filters. You'll use the metadata field label in the "field" section (ex., "ext\_orcid\_id") and the actual ID (ex., 123456) in the "value" section.

**Note:** Author ID parameters always use the operator "has-item"

Examples:

Simple filter example:

```
{"filters": [{"op": "has-item", "field": "ext_orcid_id", "value": "123456"}]}
```

Compound filter example:

```
{"filters": [{"op": "contains", "field": "publication_date", "value": "2019"}, {"op": "has-item", "field": "ext_inst_id", "value": "123456"}]}
```

You'll then use that filter to call the API as described in the filter instructions in this document.

## PUT Add an ID to a Specific Author

In the event that you add new institutional authors to your repository after your bulk spreadsheet has been processed, you can use the API to add institutional IDs on an author-by-author basis.

This will allow you to create author metadata for an author/institution ID using the bepress user ID.

When adding author IDs:

- You can add up to five external IDs, with labels defined by you.
- The API will overwrite all existing IDs with newly added IDs, so add all IDs at once; when adding additional IDs, include any previous IDs as well.
- After you add one or more IDs to an author, it may take up to an hour for metadata to be refreshed. Wait before trying to pull a snapshot that has or is filtered by added author IDs.

### Endpoint

```
/authors/YOUR_SITE_URL
```

### Method

PUT

### Body parameters

```
{"authors": [{"author_userid": AUTHOR_ID, "inst_id": "9876543", "orcid_id": "1234567"}]}
```

The “author\_userid” refers to the author’s bepress user ID. You can find this value in the “author\_userid” field in the JSON file returned for an article on which the individual is an author. Use that value to replace `AUTHOR_ID` and omit quotes (the bepress ID is stored as a number, not a string).

### Example URL request

```
https://content-out.bepress.com/v1/authors/site.institution.edu/
```

### Sample response

```
{
  "authors": [
    {
      "author_userid": AUTHOR_ID,
      "inst_id": "9876543",
      "orcid_id": "1234567"
    }
  ]
}
```

```
}
]
}
```

Results should show the same contents as you entered in the body parameters.

## GET Get Author Metadata for Specific Author ID

This allows you to confirm creation or addition of author metadata.

### Endpoint

```
/authors/YOUR_SITE_URL/AUTHOR_ID
```

Replace `AUTHOR_ID` with the bepress user ID, which you can find labeled "author\_userid" in the JSON file returned for an article on which the individual is an author.

### Method

GET

### Example request URL

```
https://content-out.bepress.com/v1/authors/site.institution.edu/999999x
```

### Sample response

```
{
  "authors": [
    {
      "author_userid": AUTHOR_ID,
      "inst_id": "9876543",
      "orcid_id": "1234567"
    }
  ]
}
```

## GET Get All Author Metadata for Your site

Retrieve a list of author metadata for all authors to whom you have added it.

### Endpoint

```
/authors/YOUR_SITE_URL/
```

## Method

GET

## Example request URL

```
https://content-out.bepress.com/v1/authors/site.institution.edu/
```

## Sample response

```
{
  "authors": [
    {
      "author_userid": AUTHOR_ID,
      "inst_id": "9876543210",
      "scopus_id": "654321"
    },
    {
      "author_userid": AUTHOR_ID_2,
      "orcid_id": "1234567",
      "inst_id": "9876543"
    }
  ]
}
```

# Common Filter Queries

Below are some typical queries for which you can create search-based filters, along with sample body parameters. For each query, follow the steps for working with filters above to first perform a PUT request to create a search-based filter, then a GET request to call results for that filter.

Results will include all metadata for each record. Responses will all appear as described under "Retrieving JSON Files."

## Metadata for Author Name in a Specific Series

An example of when you might want this data is to pull content by a particular faculty member in a given subject for display on a faculty webpage, or to add to faculty evaluation systems.

### Body parameters

```
{"filters":[{"op":"contains", "field":"authors", "value":"Amy Authorton"},{
"op":"has-item", "field": "ancestor_link", "value": "http://site.
institution.edu/biology_facpub"}]}
```

## Metadata for an ORCID ID (previously added to an author via the API)

This query is similar to the previous query, but it uses an ID instead of author name to pull records by an author. You'll first need to add an institutional ID or other external ID (such as an ORCID ID), if you haven't yet done so using the steps in the section on filtering by authors above. When you call the API to return metadata for an ID, you'll use the label you created with the addition of "ext\_" at the beginning (in this case, "ext\_orcid\_id" for the label created as "orcid\_id").

### Body parameters

```
{"filters":[{"op":"has-item", "field":"ext_orcid_id", "value":"1234567"}]}
```

## Metadata for Specific Departmental Series and Year

This may be helpful when pulling data for departmental webpages or faculty reporting systems.

### Body parameters

```
{"filters":[{"op":"contains", "field":"publication_date", "value":"2019"}, {"op":"has-item", "field": "ancestor_link", "value": "http://your institution.edu/biology_facpub"}]}
```

## Metadata for Most Recent Issue of Journal

Pull data for upload to DOAJ or CrossRef.

### Body parameters

```
{"filters": [{"op": "has-item", "field": "ancestor_link", "value": "http://site.institution.edu/journal_title"}, {"op": "contains", "field": "publication_date", "value": "2020-03"}]}
```

## Metadata for ETDs with a Specific Keyword

Review or highlight graduate research in a certain area based on a word in the metadata.

### Body parameters

```
{"filters": [{"op": "has-item", "field": "ancestor_link", "value": "http://site.institution.edu/etd"}, {"op": "contains", "field": "abstract", "value": "sleep"}]}
```

## Metadata for a Specific ETD Advisor in a Department

Find all graduate research projects with a specific advisor.

## Body parameters

```
{"filters": [{"op": "has-item", "field": "ancestor_link", "value": "http://site.institution.edu/etd"}, {"op": "equals", "field": "department", "value": "Chemistry"}, {"op": "contains", "field": "custom_field_advisor1", "value": "Andrew Advisor, Ph.D."}]}
```

## FAQ

### **What type of error codes are returned by the API?**

The Digital Commons API is built around REST and uses standard HTTP error codes.

### **Do API results display non-Latin characters?**

Yes, non-Latin characters should display as expected in the JSON file.

### **How long does it take to get the results of an API request?**

We estimate results will be returned in about 10 minutes or so. If it has been more than an hour, please call the API again. Let us know if it is taking longer than that.

### **What is the best way to locate back-end metadata field names for use in search-based filters?**

You can run a [Content Inventory](#) metadata report for your site or publication structure. At the publication level, you can also use [Batch Revise](#) to export a spreadsheet with current metadata.