ReviewDB API

Authorization scenarios

Most operations in ReviewDB require simple "write" or "read" permissions on a particular review. How these permissions are granted depends on the authorization realm for the review.

Review creation and protected meta-data

Creating reviews (other than practice reviews) and updating certain fields via the ReviewDB API requires system administrator privileges. These operations are generally expected to be done via another system (e.g. Archie or Porto) instead.

This table outlines the permissions needed to change Review fields:

System administrator	Write meta-data permission	Write permission	Maintained internally
realm unitId (Porto) ownedByIndividual (Porto) type subType status submitToEM cochraneReview practiceKey templateIndicator sourceReviewId Specific to Editorial Manager: submissionStatus invitationStatus invitationDueDate journalCode journalTitle invitationId parentDocumentId revisionNumber	title cdNumber stage reviewNo (code) phase lastPublishedDate protocolPublishedIssueNo protocolPublishedYear reviewPublishedIssueNo reviewPublishedYear lastCitationIssueNo lastCitationYear groupId Obsolete fields: documentPK tags assessedUpToDate nextStage studiesManagedBy	searchDate useStudyCentricDataStructures gradeProIntegrationEnabled riskOfBiasMethod reviewFormat	changedSinceLastTag reviewModified reviewModifiedBy titleModified titleModifiedBy

Authorization realms

Authorization is handled differently based on the "realm" setting for the review, as outlined in this table:

	ARCHIE	PORTO	PRACTICE	PRACTICE_TEMPLATE
Read	Managed by Archie, depends on review phase	Granted to users with a role on the review, with a "manage reviews" role in the owning unit, and to system administrators	Granted to user who owns the review, and to system administrators	Granted to user who owns the review, and to system administrators
Write	Managed by Archie, depends on review phase	Granted to users with a role on the review, with a "manage reviews" role in the owning unit, and to system administrators on active reviews	Granted to user who owns the review, and to system administrators	Granted to user who owns the review, and to system administrators
Meta-data	Managed by Archie	Granted to users with a "manage reviews" role in the owning unit, and to system administrators	Granted to user who owns the review, and to system administrators	Granted to user who owns the review, and to system administrators
Protected meta-data	Managed by Archie	Managed by Porto (use Porto API)	System administrators	System administrators
Create review	Managed by Archie	Managed by Porto (use Porto API)	Any user (up to a maximum number of practice reviews)	Members of the trainers' network (Archie group)

Cochrane REST API

The Cochrane Application Programming Interface (API) is based on Representational state transfer (REST) and uses resource-oriented URLs. The API **only** supports SSL – any non-SSL calls will return a 403 (FORBIDDEN) return code.

Where possible the API strives to use appropriate HTTP verbs for each action:

Verb	Description
GET	Used for retrieving resources or collection of resources

POST	Used for creating resources or collection of resources
PUT	Used to update existing resources or for collection of resources (cannot be used to create resource unless specified)
PATCH	Used to partially update existing resources or collection of resources
DELETE	Used for deleting resources or collection of resources

Cochrane API is located at https://api.cochrane.org (documentation at https://api.cochrane.org/api-docs/index.html)

For the test version of each endpoint, replace api.cochrane.org with test-api.cochrane.org. (documentation at https://test-api.cochrane.org/api-docs/index.html)

Changes

As data in Cochrane evolves so will the API. Changes to the API will be listed on the *What's New* page. It is possible to sign up for notifications via Email of any changes. We strive to send out notifications as soon as possible for Non-Backward compatible issues to allow clients to update their software. If we are required to release a new API we will support two versions for as long as possible (therefore clients should update their software to the newest version ASAP).

We strive to make as many changes backward compatible as possible. New fields will, if possible, be optional and have default values if not specified.

Current version

The current version of the API is **v1**. Cochrane REST API uses *Accept-headers* to store the version information. If omitted, **version 1 will be used**. Regularly check back here or the swagger documentation to see what is the current version of the API. Request a specific version in the *Accept-header* as shown in the examples below.

Version can be specified in three different ways:

- Application/**vnd.cochrane.v3+**json
- Application/vnd.cochrane.org+json; **version=3**
- Application/json; **version=3**

The above examples would request for version 3 of the API in *JSON* format.

If omitting the format:

- Application/**vnd.cochrane.v3**
- Application/vnd.cochrane.org; **version=3**
- Application/*; **version=3**

JSON will be used as default.

What's New

Nothing to add here yet…

Authentication

OAuth2 is used for authentication. With (almost) each request to an endPoint an OAuth token needs to be provided. The API queries Archie for permissions using the token.

The token should be sent in the Authorization-Header:

```
Authorization: Bearer "token-value"
```

If no AuthorizationHeader is sent and Authorization is expected, a 403 (FORBIDDEN) is returned.

If there is no valid AuthorizationHeader and Authorization is expected, a 401 (UNAUTHORIZED) is returned

Requests

The base URL for the Cochrane API is https://api.cochrane.org (and https://test-api.cochrane.org for testing)

Requests can be in many different formats including: *JSON*, *CSV*, *Smile*, *XML*. The type is defined in the *Content-Type-Header*. If no *Content-Type-Header* is defined – *JSON* is used as a default.

GET – Reading Data

Data from Cochrane can be read by issuing an HTTP GET request to an endpoint. To GET *Study 987* for *Review 123*, send a GET request to: https://api.cochrane.org/reviews/123/studies/987

A successful request will be indicated by a 200 (OK) HTTP status code. The response will contain the data being retrieved (this example in JSON):

```
{
    "id": "987",
    "reviewId": 123,
    "name":"Henrik",
    "year":"1973",
    "sectionId": 0,
    "dataSource": "PUB",
    "noteId": 0
}
```

If an error occurs – it will most often return a 404 (NOT FOUND)

PUT – Writing Data

Update data by issuing a HTTP PUT request to an endpoint. To update *Study 987* for *Review 123*, send a PUT request to: https://api.cochrane.org/reviews/123/studies/987

With a payload (the following example is in *JSON* format):

```
{
    "id": "987",
    "reviewId": 123,
    "name":"Henrik HL",
    "year":"2014",
    "sectionId": 4,
    "dataSource": "MIX"
}
```

Would update the *study 987*. When issuing a PUT request, the database record content will be overwritten with the data sent in the PUT request. Omitted fields will be cleared.

A successful request will be indicated by a 200 (OK) HTTP status. The response will contain the data written (here in JSON):

```
{
    "id": "987",
    "reviewId": 123,
    "name":"Henrik HL",
    "year":"2014",
    "sectionId": 4,
    "dataSource": "MIX"
}
```

NB: Don't use PUT to create new records even though the client provides the ID – instead use POST (weather or not POST accepts client provided ID's depends on the endPoint).

In the case of an error – it will most often return a 404 (NOT FOUND) or 400 (BAD REQUEST)

PATCH – Updating part of the data

To partially update a record issue a PATCH request to an endpoint. To partially update *Study 987* for *Review 123*, send a PATCH request to: https://api.cochrane.org/reviews/123/studies/987

With a payload (the following example is in *XML* format):

```
<Study>
    <id>987</id>
    <reviewId>123</reviewId>
    <name>HHL</name>
    <year>2010</year>
</Study>
```

This will partially update *study 987* with (only) the name and year field. The name will be updated to HHL and the year will be updated to 2010, the rest of the fields in the record would remain unchanged.

NB: It is **NOT** possible to send default values (*Null, 0, false, etc*) using the PATCH request. All fields containing default values will be ignored. In the case where a field needs to be updated with a default value – use PUT.

POST – Creating Data

Create a new record with a POST request to an endpoint. To create a new Study for *Review 123*, POST a request to: https://api.cochrane.org/reviews/123/studies

With a payload (the following example is in XML format):

```
<Study>
    <name>HHL</name>
    <year>2015 </year>
    <sectionId>5</sectionId>
    <dataSource>PUB</dataSource>
</Study>
```

This action would create a new Study record. The result would look like this (in XML format):

```
<Study>
  <id>4568108645260278276</id>
  <name>HHL</name>
  <year>2015</year>
  <sectionId>5</sectionId>
  <dataSource>PUB</dataSource>
  <reviewId>123</reviewId>
  <noteId>0</noteId>
</Study>
```

In the case of an error – it will most often return a 400 (BAD REQUEST).

NB: Use POST with client provided ID's (only a few endPoints support client provided ID's)

DELETE – Removing Data

Delete data with a DELETE request. To delete *Study 4568108645260278276* for *Review 123*, send a DELETE request to: https://api.cochrane.org/reviews/123/studies/4568108645260278276

A successful delete request will be indicated by a 204 (OK) status code with an empty response. If an error occurs it will most likely be a 404 (NOT FOUND).

OPTIONS - endPoint information

OPTIONS is available on all endPoints. It is mainly implemented for the purpose of CORS (**C**ross-**o**rigin **r**esource **s**haring). When sending an OPTIONS to any andPoint a 200 (OK) will be returned with the following headers:

- **Access-Control-Allow-Origin** - returns a list of sources that can access the API. Currently accessable by all ("*****" is returned)

- **Access-Control-Allow-Headers** - returns a list of headers allowed. Currently returning: *Authorization*, *Content-Type*, *Accept*, *If-None-Match*, *If-Modified-Since*
- **Access-Control-Allow-Methods** - returns a list of REST methods allowed for the specific endPoint. This is a comma separated list, eg: OPTIONS,*GET,POST,PUT,PATCH,DELETE. Depending on the endPoint, 1-6 methods will be listed*
- **Access-Control-Max-Age** - returns number of seconds the response should cache values (current default value is *3600*)
- **Access-Control-Expose-Headers** - returns a list of exposed headers. Currently returning: *Content-Type*, *Cache-Control*, *Link*, *Total-Count*, *ETag*

Rate Limiting

For all requests a *Rate-Limit* might be inserted in the future. If your *Rate-Limit* has been reached (the maximum number of requests per minute/hour) a 429 (TOO MANY REQUESTS) response will be returned. In this case try again after a short while.

Conditional requests

All responses return an *ETag-header*. Almost all responses also return a *Last-Modified-header*. You can use the values of these headers to make subsequent *GET* requests to those resources using the *If-None-Match* and *If-Modified-Since-headers*, respectively. If the resource has not changed, the server will return a 304 (NOT MODIFIED).

NB: making a conditional request and receiving a 304 (NOT MODIFIED) response does not count against your Rate Limit, so we encourage you to use it whenever possible. This will help on server loads.

Language

The language is specified in the request header by using the *Accept-language* header. If not specified, the default language is the default language of the request. Currently only English is supported. *Accept-language* is primarily for error messages

Accept-Language: da-dk

In the above example, the error message will be supplied in Danish if available. In the case where the language is not supported – English will be used.

Parameters

The Cochrane REST API accepts the following additional query parameters and values. URL query-string parameters are used for filtering – not for resource names.

Download

If you would like to trigger a file download of your data from a web browser, add *download=XXX*. This will cause the REST service to add the appropriate headers so that browsers know to save the data to a file (called 'XXX').

https://api.cochrane.org/authors/333?**download=henrik.xml**

The above example will request for the *author 333* and trigger a download in the browser. The format of the file (the response) is set in the *Accept-Type* header.

The *download* parameter can be used with all request methods: *GET*, *PUT*, *POST*, *PATCH* but not *DELETE*.

If the *download* parameter is used in a *GET* request with the *ETag* matching – the *download* parameter will supersede the 304 (NOT MODIFIED) and trigger a download even though the data has not been changed – thus resulting in a 2XX status code.

Envelope

envelope=true – In some situations it is necessary to have the http status codes enveloped inside the result. If set the result will be enveloped with HTTP result code, status, a message and the result wrapped in a data element:

- **code** – contains the HTTP response status code as an integer.
- **status** – contains the text: "success", "fail", or "error". Where "fail" is for HTTP status response values from 500-599, "error" is for statuses 400-499, and "success" is for everything else (e.g. 1XX, 2XX and 3XX responses).
- **message** – only used for "fail" and "error" statuses to contain the error message in English. If Accept-Language header is set and language is supported the error message will be supplied in the requested language.
- **data** – contains the response body. In the case of "error" or "fail" statuses, this contains the cause, or exception name.

An example of an *envelope*:

https://api.cochrane.org/reviews/123/studies/987?envelope=true

If successful, this would return (in *JSON* format):


```
{
  "code":200,
  "status":"success",
```

```
  "data":{
    "id": "987",
    "reviewId": 123,
    "name":"Henrik",
    "year":"1973",
    "sectionId": 0,
    "dataSource": "PUB",
    "noteId": 0
  }
}
```

EdgedTitleLinks

When requesting for collections of items which don't fit within a paged result it is nice to know the boundaries for other pages. Eg. when you request for page 15 in a list of Studies for a review, the result is the Studies on that page. When turning *edgedTitleLinks* on (*edgedTitleLinks=true*) all other generated link-header-pages will contain (in their title) the boundaries: "From which Study - to which Study" does the page contain. So if getting all Studies for a review results in 20 pages, the current page is 15 and the pages requested (see pages under the pagination section) is *-2,-1,1,2* then the result could look something like this (only displaying link headers):

```
<https://api.cochrane.org/reviews/123/studies?page=13>; rel="-2"; page="13"; title="ISRCTN73545489 - Karner 2012",

<https://api.cochrane.org/reviews/123/studies?page=14>; rel="-1"; page="14"; title="Katz 2004 - Kourouklis 1976",

<https://api.cochrane.org/reviews/123/studies?page=16>; rel="1"; page="16"; title="Lieberman 1988 - Matzneller 2013",

<https://api.cochrane.org/reviews/123/studies?page=17>; rel="2"; page="17"; title="Mazeh 2006 - Moore 1980"
```

Excluded

excluded=id,name,... As an optimization, you can specify which fields not to return. Field names are separated by a comma.

https://api.cochrane.org/reviews/123/studies/987?excluded=name,year,versionStart,versionEnd

The above request would return:

```
{
    "id": "987",
    "reviewId": 123,
    "sectionId": 0,
    "dataSource": "PUB",
    "noteId": 0,
    "createdBy": "anonymous",
    "deletedBy": null,
    "versionNo": 5
}
```

NB: It is not possible to use the *excluded* parameter in combination with Included. If both query parameters are used an error will 400 (BAD REQUEST) will be returned.

NBB: As a default *versionStart*, *versionEnd*, *createdBy*, *deletedBy* and *versionNo* are excluded from every result. If these fields are required then provide an empty excluded (https://api.cochrane.org/reviews/123/studies/987?excluded).

Expand

Many endpoints support expanding the related resources to minimize the required number of API round trips. It is used by supplying a comma separated list of Resources. Each Resource lists other Resources that are expandable.

https://api.cochrane.org/reviews/123/studies?expand=reference

The above example would retrieve a collection of all studies for *review 123*. An object representing the reference(s) will be inside each Study in the collection.

To further manipulate the expanded result, append a '**[**' and '**]**' to the end of the field to expand. Manipulation could be: *further expand*, *sorting* and/or *filtering* . *expand, sorting* and *filtering* can be used in any combination and at any level

NB: Please keep in mind that this can be a *very CPU and bandwidth intensive feature* - use with care. Once Rate Limiting is implemented - use of Expand will be an expensive call.

Nested Expand

It is allowed to make nested expands:

https://api.cochrane.org/reviews/123/studies?expand=reference[expand=referenceIdentifiers]

In the above example all referenceIdentifiers to the references will be expanded (and returned) within the references (which is returned within the Studies). There is no limit to the number/levels of expands.

sort

Within an expand it is also possible to sort the result:

https://api.cochrane.org/reviews/123/studies?expand=reference[sort=name]

The above example will sort the reference-list within each Study by name. Sorting in expand uses the same rules as sorting for collection results.

filter

Within an expand it is also possible to filter the result:

https://api.cochrane.org/reviews/123/studies?expand=reference[q=hello]

The above example will only return expanded references on Studies for review 123 that contain the String "hello". Filtering in expand uses the same rules a filtering for collection results.

expand

It is further possible to expand an expanded result - see Expand.

expandAll

It is possible at any point to use expandAll to expand recursively - see ExpandAll.

hideEmptyList

In order not to send empty result-lists - it is possible to set hideEmptyList=true - this will prevent empty lists from being sent.

ExpandAll

ExpandAll is a shortcut in order to expand everything on a Resource. In order not to create circular expands cirtain limitations are inserted when using ExpandAll. When expandAll is used without a value - all fields below the resource is recursively expanded. In order just to expand a list of fields this can be done by writing a comma separated list of fieldNames:

https://api.cochrane.org/reviews/123/textSections?expandAll=textSections,note

The above expands all textSections of review 123 recursively + all notes of all textSections.

It is possible to further manipulate the expandAll result, append a '**[**' and '**]**' to the end of the field just like Expand.

Hal

Turn hal on (=*true*) or off (=*false*). This query parameter will supersede the *Accept-header*. If turned on the result will include hal information.

Included

included=id,name,… As an optimization, you can specify which fields to return. Field names are separated by a comma.

https://api.cochrane.org/reviews/123/studies/987?included=name,year

The above request would return:

{

"name":"Henrik",

"year":"1973"

}

NB: It is not possible to use the *included* parameter in combination with *Excluded*.

Metadata

Only used for requests that return a collection. For all requests (*GET*, *PUT*, *PATCH*, *POST* and *DELETE*) it sends the *total_count* – which is the number of items returned.

```
"_metadata": {
    "total_count": 2
}
```

In case of a *GET* request also: current *page*, number of items *per_page* and total number of *pages* are returned.

```
"_metadata": {
    "page": 1,
    "per_page": 30,
    "pages": 1,
    "total_count": 2
}
```

Pagination

A *GET* request that returns a collection will be set to *30 items per page* by default. (The maximum number of items per page is 100). If a different number of items per page is required – use the *per_page* query parameter. Use the *page* parameter to specify the page wanted. Use the pages parameter to create user defined Link-headers.

The page numbering is 1-based and omitting the *page* parameter would result in the first page. If no pagination is specified and the result is bigger than the number of items per page (30 items if nothing is specified) a 206 (PARTIAL CONTENT) will be returned. A 206 (PARTIAL CONTENT) will also be returned if the requested *page* and *per_page* cannot be fulfilled (in case you request the last page and it does not contain enough items to fill the *per_page* specified).

page=3 – would get the 3rd page of results (by default this would be a result of items 61-90)

per_page=40 – the number of results is set to 40 per page (page 1 contains items 1-40, page 2 contains items from 41-80, etc)

pages=-2,-1,1,2 - would create Link-headers for pages -2, -1, 1 and 2 relative to the current page. pages supports ranges. In order to specify a range write: [xxx TO yyy]. In this case Links to pages xxx to page yyy will be generated. xxx must be smaller than yyy otherwise a 400 (BAD REQUEST) will be thrown. A combination of ranges and single pages is supported: *pages=-10,[-5 TO 5],10*

In all collection results a response header: *Total-count* is added containing the total number of items in the result.

Link Header

The pagination info is included in the *Link-header*. Use these links instead of creating your own.

```
Link: <https://api.cochrane.org/reviews/123/studies?page=3&per_page=20>; rel="next"
```

The above link would point to the *next* page which in this case would be: https://api.cochrane.org/reviews/123/studies?page=3&per_page=20

Below is a list of possible values in the *rel* (=relative) part of the link above.

Name	Description
next	Shows the URL of the immediate next page of results.
last	Shows the URL of the last page of results.
first	Shows the URL of the first page of results.
prev	Shows the URL of the immediate previous page of results.
self	Shows the URL to this result/resource (can also contain a *title*).

Link-headers are added if available. On non-collection results – a title will be provided with the “self” link.

Print

print=pretty - If you’re debugging your application, you may find it useful to view the data in a human-readable format.

https://api.cochrane.org/reviews/123/studies/987?print=pretty

print=silent - As an optimization you can add *print=silent* to suppress the output from the server when writing data. The response will be empty and indicated by a 204 (NO CONTENT) if successful.

https://api.cochrane.org/reviews/123/studies/987?print=silent

Filter

It is possible to filter a collection result further. To do this use q=. q= supports different kinds of filters:

q={searchstring} – simple searching can be done using the parameter ***q={searchstring}***. *q=Henrik* will further filter the result, only returning Resources that contain the word “Henrik”.

https://api.cochrane.org/reviews/123/studies/?q=Henrik

This will return all *studies* for the *review 123* containing a field (any field) with the literal string “Henrik” (case sensitive). When searching for numbers they are converted to strings for matching - thus making it possible to search in the middle of a number/the start of a number or the end of a number: 15 would match the numbers *2015*, *1505*, as well as any string containing 15.

q={field}:{searchstring} - this narrows the filtering to the field by the name of {field}. It will only match {searchstring} if it is inside {field}. In this case {searchstring} is NOT casesensitiv.

https://api.cochrane.org/reviews/123/studies/?q=name:Henrik

This will return all Studies in review 123 containing the string "henrik" (case insensitive) in the name field.

q=_exists_:{fieldName} - will filter the result listing all that have the field with the name {fieldName} set (where the field value is **NOT** Null)

https://api.cochrane.org/reviews/123/studies/?q=_exists_:year

This will return all Studies in review 123 that doesn't contain Null in the year field

q=_missing_:{fieldName} - will filter the result listing all that don't have the field with the name{fieldName} set (where the field value **is** Null)

https://api.cochrane.org/reviews/123/studies/?q=_missing_:year

This will return all Studies in review 123 that contain Null in the year field

At the moment there is **NO** support for boolean operators or parenthesis. Only one q= will be used if multiple filters are listed in the url. More filters will be added.

NB: Use this option with caution as it is NOT a database search. The collection returned is sequentially filtered by looking at each field in each item in the collection. Once Rate Limiting is implemented - use of Filter will be an expensive call.

Sort

It is possible to sort a collection of results. When sorting is required, specify a comma separated list of field names to sort by. If a “-“ is used before the field name the collection is sorted in descending order.

https://api.cochrane.org/reviews/123/authors?sort=lastName,-firstName

This will sort the collection of *authors* in the *review 123* by first *lastName* in ascending order and then by *firstName* in descending order.

In expanded results, it is possible to specify a path for sorting. This would sort a collection based on values in the expanded result. The expanded result MUST be a single resource, it can not be a collection result:

https://api.cochrane.org/reviews/123/outcomes?expand=outcomeAnalysis&sort=outcomeAnalysis/dataType

This would get all Outcomes for review 123 and sort them on the dataType provided in the outcomeAnalysis expanded on the outcome. In this example it is also possible the other way around:

https://api.cochrane.org/reviews/123/outcomesAnalysis?expand=outcome&sort=outcome/name

This would get all OutcomeAnalysis for review 123 and sort them on the name provided in the expanded outcome.

Responses

The default encoding for responses is *HAL+JSON*. If a different encoding is wanted, it must be specified in the *Accept-header*. The following types are accepted:

Type	Header	Description
HAL+JSON	Application/hal+json	Returns JSON enclosed with HAL

HAL+XML	Application/hal+xml	Returns XML enclosed with HAL
JSON	Application/json	Returns JSON
XML	Application/xml	Returns XML
CSV	Text/csv	Returns CSV
Smile	Application/x-jackson-smile	Returns Smile

If multiple Accept-headers are in the response – the API will prioritize the Accept-header closest to the top of the above list.

HAL

HAL (or **H**yperMedia **A**pplication **L**anguage) provides a set of conventions for expressing hyperlinks in either *JSON* or *XML*.

More to come – see: http://stateless.co/hal_specification.html

Hal example for a study (in *JSON* format):

```
"_links": {
        "self": {
            "href": "https://api.cochrane.org/reviews/123/studies/10167072993706414l2",
            "title": "HHL"
        }
}
```

Etc…

HTTP Status Codes

Cochrane uses conventional HTTP response codes whenever possible to indicate success or failure of an API request.

Success codes

Code	Title	Description
200	OK	Request succeeded. Response included
201	Created	Resource created. URL to new resource in Location header
204	No Content	Request succeeded, but no response body
206	Partial Content	Result is too big - typical for results containing collections
304	Not Modified	The request has not been modified since the last request

Error codes

Code	Title	Description
400	Bad Request	Could not parse request
401	Unauthorized	No authentication credentials provided or authentication failed
403	Forbidden	Authenticated user does not have access
404	Not Found	Resource not found
415	Unsupported Media Type	The Media type supplied in the Accept header is not supported
422	Unprocessable Entry	A request to modify or create a resource failed due to a validation error
429	Too Many Requests	Request rejected due to rate limiting
500-599		An internal server error occurred

Entry Points

All endPoints are documented in Swagger on the following url:

https://test-api.cochrane.org/api-docs

All endPoints are camelCased. For all endPoints integrity checks are made to ensure valid data.