title: o2r web API v1.0
language_tabs:

- shell: Shell
- http: HTTP
- javascript: JavaScript
- ruby: Ruby
- python: Python
- php: PHP
- java: Java
- go: Go
  toc_footers:
- Find more info in our documentation.
  includes: []
  search: true
  highlight_theme: darkula
  headingLevel: 2

o2r web API v1.0

> Scroll down for code samples, example requests and responses. Select a language for code samples from the tabs above or the mobile navigation menu.

About

The o2r web API acts as the interface between the o2r microservices and the web interface.

The API provides services around the executable research compendium (ERC), or "compendium" for short, which is documented in the ERC spec.

A good starting point for understanding the different parts of the API is the compendium life-cycle.
The API is implemented as a RESTful API. The entrypoint for the current version is /api/v1.
Unless specified otherwise, responses are always in JSON format. Body parameters in `POST` requests are expected in `multipart/form-data format.` Requests to the API should always be made with a secure connection using `HTTPS`. Some requests require authentication with a specific user level.

To cite this specification please use

> *Nüst, Daniel, 2018. Reproducibility Service for Executable Research Compendia: Technical Specifications and Reference Implementation. Zenodo. doi:10.5281/zenodo.2203844*

For a complete list of publications, posters, presentations, and software projects from the o2r project please visit https://o2r.info/results/.

License

Compendium lifecycle

An authorized user (see user levels) can create compendia from different sources, i.e. direct API upload or via a public share.
The uploaded workspace is then treated as a candidate compendium until all required metadata is provided.
A candidate must be saved to become publicly available, and then may not become a candidate again.
A published compendium can be executed and the whole compendium can be shipped to repositories for preservation purposes.
A user can combine data and workflow of multiple compendia with substitution.

Specific users can (i) create public links for candidates so these can be inspected by third parties without logging in, and (ii) manage users).

The API provides a generic search endpoint for finding compendia and jobs and uses ORCID and OAuth for authentication.

Base URLs:

- https://o2r.uni-muenster.de/api/v1

Email: o2r project Web: o2r project

Authentication

- API Key (cookie_authenification)
  - Parameter Name: **connect.sid**, in: cookie.

Compendium

Candidate process

After uploading a compendium is *not* instantly publicly available.
It is merely a **candidate**, because metadata still must be completed for the compendium to be valid.
Users with sufficient rights can create public links for a candidate compendium.

The following process models this intermediate state of a compendium.

Creation and view

Candidates can be identified by the property `candidate`.
It is set to `true` after creating a new compendium by upload or public share submission **and** the authoring user having reviewed the metadata.

⚠ Note

It is not possible to circumvent the metadata review. Only a successful metadata update can set `candidate: true`.

Example:

```
{
  "id":"12345",
  "metadata": … ,
  "created": "2016-08-01T13:57:40.760Z",
  "files": …,
  "candidate": true
}
```

Only the creating user and users with required level can view a candidate and see the `candidate` property while it is `true`.

When accessing a list of compendia for a specific user *as that user*, then this list is extended by available candidates.
The candidates may be added to the response independently from any pagination settings, i.e. if a client requests the first 10 compendia for a user having two candidates, the client should be prepared to handle 12 items in the response.

Metadata review and saving

After the user has reviewed and potentially updated the metadata as required and saved them successfully, then the candidate status is changed (`candidate: false`) and the compendium is publicly available.

The `candidate` property is not exposed any more if it is `false`.

It is *not* possible to save invalid metadata or to manually change the `candidate` property, therefore a compendium cannot become a candidate again after successful completion of the creation.

Deletion

Unlike published compendia, a candidate can be deleted by a the authoring user, see delete compendium.

upload_compendium

> Code samples

```
# You can also use wget
curl -X POST https://o2r.uni-muenster.de/api/v1/compendium \
  -H 'Content-Type: multipart/form-data' \
  -H 'Accept: application/json'
```

```
POST https://o2r.uni-muenster.de/api/v1/compendium HTTP/1.1
Host: o2r.uni-muenster.de
Content-Type: multipart/form-data
Accept: application/json
```

```
const inputBody = '{
  "content_type": "compendium",
  "compendium": "string",
  "path": "string",
  "doi": "string",
  "zenodo_record_id": "string",
  "filename": "string"
}';
const headers = {
  'Content-Type':'multipart/form-data',
  'Accept':'application/json'
};

fetch('https://o2r.uni-muenster.de/api/v1/compendium',
```

```
{
  method: 'POST',
  body: inputBody,
  headers: headers
})
.then(function(res) {
    return res.json();
}).then(function(body) {
    console.log(body);
});
```

```
require 'rest-client'
require 'json'

headers = {
  'Content-Type' => 'multipart/form-data',
  'Accept' => 'application/json'
}

result = RestClient.post 'https://o2r.uni-muenster.de/api/v1/compendium',
  params: {
  }, headers: headers

p JSON.parse(result)
```

```
import requests
headers = {
  'Content-Type': 'multipart/form-data',
  'Accept': 'application/json'
}

r = requests.post('https://o2r.uni-muenster.de/api/v1/compendium', headers = headers)

print(r.json())
```

```
<?php

require 'vendor/autoload.php';

$headers = array(
    'Content-Type' => 'multipart/form-data',
    'Accept' => 'application/json',
);

$client = new \GuzzleHttp\Client();

// Define array of request body.
$request_body = array();

try {
    $response = $client->request('POST','https://o2r.uni-muenster.de/api/v1/compendium', array(
        'headers' => $headers,
        'json' => $request_body,
       )
    );
    print_r($response->getBody()->getContents());
 }
 catch (\GuzzleHttp\Exception\BadResponseException $e) {
    // handle exception or api errors.
    print_r($e->getMessage());
 }

 // ...
```

```
URL obj = new URL("https://o2r.uni-muenster.de/api/v1/compendium");
HttpURLConnection con = (HttpURLConnection) obj.openConnection();
con.setRequestMethod("POST");
int responseCode = con.getResponseCode();
BufferedReader in = new BufferedReader(
    new InputStreamReader(con.getInputStream()));
String inputLine;
StringBuffer response = new StringBuffer();
while ((inputLine = in.readLine()) != null) {
    response.append(inputLine);
}
in.close();
System.out.println(response.toString());
```

```
package main

import (
       "bytes"
       "net/http"
)

func main() {

    headers := map[string][]string{
        "Content-Type": []string{"multipart/form-data"},
        "Accept": []string{"application/json"},
    }

    data := bytes.NewBuffer([]byte{jsonReq})
    req, err := http.NewRequest("POST", "https://o2r.uni-muenster.de/api/v1/compendium", data)
    req.Header = headers

    client := &http.Client{}
    resp, err := client.Do(req)
    // ...
}
```

`POST /compendium`

Upload via API

Upload a research workspace or full compendium as a compressed `.zip` archive with an HTTP `POST` request using `multipart/form-data`.

The upload is only allowed for logged in users. Upon successful extraction of archive and processing of the contents, the `id` for the new compendium is returned.

⚠ Required user level and authentication

The user creating a new compendium must have the required user level. Requests must be authenticated with a cookie `connect.sid`, see user authentication.

```
curl -F "compendium=@compendium.zip;type=application/zip" \
    -F content_type=compendium \
    --cookie "connect.sid=<cookie string here>" \
    https://…/api/v1/compendium
```

```
curl -F "compendium=@path/to/workspace.zip;type=application/zip" \
    -F content_type=workspace \
    --cookie "connect.sid=<cookie string here>" \
  https://…/api/v1/compendium
```

⚠ Important

After successful upload the candidate process must be completed for workspaces.

For local testing you can quickly upload some of the example compendia and workspaces from the erc-examples project.

Public share

Load a research compendium by submitting a link to a cloud resource using an HTTP `POST` request using `multipart/form-data`.

Currently, the following repositories are supported:

- Sciebo
- Zenodo
- Zenodo Sandbox

Common

All repositories use the same API endpoint `https://…/api/v1/compendium`, but with different required/optional parameters.

The upload is only allowed for logged in users.

⚠ Required user level and authentication

The user creating a new compendium must have the required user level. Requests must be authenticated with a cookie `connect.sid`, see user authentication.

To run the load from the command line, login on the website and open you browser cookies.
Find a cookie issued by `o2r.uni-muenster.de` with the name `connect.sid`.
Copy the contents of the cookie into the request example below.

Upon successful download from the public share, the `id` for the new compendium is returned.

```
curl -F "content_type=compendium" \
    -F "share_url=https://uni-muenster.sciebo.de/index.php/s/G8vxQ1h50V4HpuA"  \
    --cookie "connect.sid=<code string here>" \
     https://…/api/v1/compendium
```

⚠ Warning "Important"

After successful load from a public share, the candidate process applies.

Sciebo

Sciebo is a cloud storage service at North Rhine-Westphalian universities.
Although it builds on ownCloud and the implementation might be able to handle any ownCloud link, only Sciebo's publish shares are supported by this API.

File selection

Depending on the public share contents different processes are triggered:

1. If a file named `bagit.txt` is found, the directory is checked for Bagit validity
2. If a single zip file is found, the file is extracted, if multiple zip files are found, the filename has to be specified, otherwise an error is returned
3. If a single subdirectory is found, the loader uses that subdirectory as the base directory for loading
4. Depending on the value of `content_type` (see below), the public share contents are treated as a complete compendium or as a workspace

Example data

For testing purposes you can use the following public share, which contains a few ready-to-use compendia:

Example data

For testing purposes you can use the following public shares.
They contain the a compendium with metadata.

- Sciebo: `https://uni-muenster.sciebo.de/index.php/s/G8vxQ1h50V4HpuA`
- Zenodo: `https://sandbox.zenodo.org/record/69114`

> Body parameter

```
content_type: compendium
compendium: string
path: string
doi: string
zenodo_record_id: string
filename: string
```

Parameters

Name	In	Type	Required	Description
body	body	new_compendium	true	The Compendium to be uploaded.

Detailed descriptions

body: The Compendium to be uploaded.

> Example responses

> upload response

```
{
  "id": "a4Nd1"
}
```

> Bad Request due to not supported content_type

```
{
  "error": "provided content_type not implemented"
}
```

> Unauthorized due to user not being logged in.

```
{
  "error": "user is not authenticated"
}
```


> Forbidden due to user level not allowing upload


```
{
  "error": "user level does not allow compendium creation"
}
```

```
{
  "error": "host is not allowed"
}
```


> Unprocessable Entity


```
{
  "error": "public share URL is invalid"
}
```

```
{
  "error": "DOI is invalid"
}
```

```
{
  "error": "files with unsupported encoding detected: [{'file':'/tmp/o2r/compendium/ejpmi/data/test.txt','enc
}
```


Responses

Status	Meaning	Description	Schema
200	OK	upload response	upload_response
400	Bad Request	Bad Request due to not supported content_type	general_error_model
401	Unauthorized	Unauthorized due to user not being logged in.	general_error_model
403	Forbidden	Forbidden due to user level not allowing upload	general_error_model
422	Unprocessable Entity	Unprocessable Entity	general_error_model

To perform this operation, you must be authenticated by means of one of the following methods: cookie_authenification

list_compendium

> Code samples

```
# You can also use wget
curl -X GET https://o2r.uni-muenster.de/api/v1/compendium \
  -H 'Accept: application/json'
```

```
GET https://o2r.uni-muenster.de/api/v1/compendium HTTP/1.1
Host: o2r.uni-muenster.de
Accept: application/json
```

```
const headers = {
  'Accept':'application/json'
};

fetch('https://o2r.uni-muenster.de/api/v1/compendium',
{
  method: 'GET',

  headers: headers
})
.then(function(res) {
    return res.json();
}).then(function(body) {
    console.log(body);
});
```

```
require 'rest-client'
require 'json'

headers = {
  'Accept' => 'application/json'
}

result = RestClient.get 'https://o2r.uni-muenster.de/api/v1/compendium',
  params: {
  }, headers: headers

p JSON.parse(result)
```

```
import requests
headers = {
  'Accept': 'application/json'
}

r = requests.get('https://o2r.uni-muenster.de/api/v1/compendium', headers = headers)

print(r.json())
```

```
<?php

require 'vendor/autoload.php';

$headers = array(
    'Accept' => 'application/json',
);

$client = new \GuzzleHttp\Client();

// Define array of request body.
$request_body = array();

try {
```

```
    $response = $client->request('GET','https://o2r.uni-muenster.de/api/v1/compendium', array(
        'headers' => $headers,
        'json' => $request_body,
       )
    );
    print_r($response->getBody()->getContents());
 }
 catch (\GuzzleHttp\Exception\BadResponseException $e) {
    // handle exception or api errors.
    print_r($e->getMessage());
 }

 // ...
```

```
URL obj = new URL("https://o2r.uni-muenster.de/api/v1/compendium");
HttpURLConnection con = (HttpURLConnection) obj.openConnection();
con.setRequestMethod("GET");
int responseCode = con.getResponseCode();
BufferedReader in = new BufferedReader(
    new InputStreamReader(con.getInputStream()));
String inputLine;
StringBuffer response = new StringBuffer();
while ((inputLine = in.readLine()) != null) {
    response.append(inputLine);
}
in.close();
System.out.println(response.toString());
```

```
package main

import (
       "bytes"
       "net/http"
)

func main() {

    headers := map[string][]string{
        "Accept": []string{"application/json"},
    }

    data := bytes.NewBuffer([]byte{jsonReq})
    req, err := http.NewRequest("GET", "https://o2r.uni-muenster.de/api/v1/compendium", data)
    req.Header = headers

    client := &http.Client{}
    resp, err := client.Do(req)
    // ...
}
```

`GET /compendium`

List compendium

Returns up to 100 results by default.

Parameters

Name	In	Type	Required	Description
job_id	query	string	false	Comma-separated list of related job ids to filter by.
user	query	string	false	Public user identifier to filter by.

Name	In	Type	Required	Description
doi	query	string	false	A DOI to filter by.
start	query	integer	false	Starting point of the result list. `start - 1` results are skipped. Defaults to `1`.
limit	query	integer	false	Limits the number of results in the response. Defaults to `100`.

> Example responses

> Returns all the matching compendia as an array

```
{
  "results": [
    "nkm4b",
    "nb2sm",
    "..."
  ]
}
```

> If there is no compendium found, the service returns an empty list.\
> `GET /api/v1/compendium?doi=not_a_doi`

```
{
  "results": []
}
```

Responses

Status	Meaning	Description	Schema
200	OK	A json Array with the matching results.	list_response

This operation does not require authentication

view_compendium

> Code samples

```
# You can also use wget
curl -X GET https://o2r.uni-muenster.de/api/v1/compendium/{compendium_id} \
  -H 'Accept: application/json'
```

```
GET https://o2r.uni-muenster.de/api/v1/compendium/{compendium_id} HTTP/1.1
Host: o2r.uni-muenster.de
Accept: application/json
```

```
const headers = {
  'Accept':'application/json'
};
```

```
fetch('https://o2r.uni-muenster.de/api/v1/compendium/{compendium_id}',
{
  method: 'GET',

  headers: headers
})
.then(function(res) {
    return res.json();
}).then(function(body) {
    console.log(body);
});
```

```
require 'rest-client'
require 'json'

headers = {
  'Accept' => 'application/json'
}

result = RestClient.get 'https://o2r.uni-muenster.de/api/v1/compendium/{compendium_id}',
  params: {
  }, headers: headers

p JSON.parse(result)
```

```
import requests
headers = {
  'Accept': 'application/json'
}

r = requests.get('https://o2r.uni-muenster.de/api/v1/compendium/{compendium_id}', headers = headers)

print(r.json())
```

```
<?php

require 'vendor/autoload.php';

$headers = array(
    'Accept' => 'application/json',
);

$client = new \GuzzleHttp\Client();

// Define array of request body.
$request_body = array();

try {
    $response = $client->request('GET','https://o2r.uni-muenster.de/api/v1/compendium/{compendium_id}', array
        'headers' => $headers,
        'json' => $request_body,
       )
    );
    print_r($response->getBody()->getContents());
 }
 catch (\GuzzleHttp\Exception\BadResponseException $e) {
    // handle exception or api errors.
    print_r($e->getMessage());
 }

 // ...
```

```
URL obj = new URL("https://o2r.uni-muenster.de/api/v1/compendium/{compendium_id}");
HttpURLConnection con = (HttpURLConnection) obj.openConnection();
```

```
con.setRequestMethod("GET");
int responseCode = con.getResponseCode();
BufferedReader in = new BufferedReader(
    new InputStreamReader(con.getInputStream()));
String inputLine;
StringBuffer response = new StringBuffer();
while ((inputLine = in.readLine()) != null) {
    response.append(inputLine);
}
in.close();
System.out.println(response.toString());
```

```
package main

import (
       "bytes"
       "net/http"
)

func main() {

    headers := map[string][]string{
        "Accept": []string{"application/json"},
    }

    data := bytes.NewBuffer([]byte{jsonReq})
    req, err := http.NewRequest("GET", "https://o2r.uni-muenster.de/api/v1/compendium/{compendium_id}", data)
    req.Header = headers

    client := &http.Client{}
    resp, err := client.Do(req)
    // ...
}
```

`GET /compendium/{compendium_id}`

View a single compendium and metadata

This includes the complete metadata set, related job ids and a tree representation of the included files. The created timestamp refers to the upload of the compendium. It is formated as RFC-3339.

`curl https://…/api/v1/$ID`

Parameters

Name	In	Type	Required	Description
compendium_id	path	string	true	Id of the compendium

> Example responses

> Returns a object with informations about the matching compendium.

```
{
  "id": "comid",
  "metadata": "...",
  "created": "2016-08-01T13:57:40.760Z",
  "user": "0000-0001-0273-7906",
  "files": "...",
  "bag": true,
  "compendium": true,
```

```
  "supstituted": true
}
```

> The metadata for an example compendium

```
{
  "id": "12345",
  "metadata": {
    "raw": {
      "title": "Programming with Data. Springer, New York, 1998. ISBN 978-0-387-98503-9.",
      "author": "John M. Chambers",
      "more content": "..."
    },
    "o2r": {
      "title": "Programming with Data",
      "creators": [
        {
          "name": "John M. Chambers"
        }
      ],
      "publication_date": 1998,
      "more content": "..."
    },
    "zenodo": {}
  },
  "created": "…",
  "files": "…"
}
```

> Returns an error

```
{
  "error": "no compendium with this id"
}
```

Responses

Status	Meaning	Description	Schema
200	OK	Returns the matching Compendium.	view_response
404	Not Found	No compendium with the given id found.	general_error_model

This operation does not require authentication

delete_compendium

> Code samples

```
# You can also use wget
curl -X DELETE https://o2r.uni-muenster.de/api/v1/compendium/{compendium_id} \
  -H 'Accept: application/json'
```

```
DELETE https://o2r.uni-muenster.de/api/v1/compendium/{compendium_id} HTTP/1.1
Host: o2r.uni-muenster.de
Accept: application/json
```

```
const headers = {
  'Accept':'application/json'
};

fetch('https://o2r.uni-muenster.de/api/v1/compendium/{compendium_id}',
{
  method: 'DELETE',

  headers: headers
})
.then(function(res) {
    return res.json();
}).then(function(body) {
    console.log(body);
});
```

```
require 'rest-client'
require 'json'

headers = {
  'Accept' => 'application/json'
}

result = RestClient.delete 'https://o2r.uni-muenster.de/api/v1/compendium/{compendium_id}',
  params: {
  }, headers: headers

p JSON.parse(result)
```

```
import requests
headers = {
  'Accept': 'application/json'
}

r = requests.delete('https://o2r.uni-muenster.de/api/v1/compendium/{compendium_id}', headers = headers)

print(r.json())
```

```
<?php

require 'vendor/autoload.php';

$headers = array(
    'Accept' => 'application/json',
);

$client = new \GuzzleHttp\Client();

// Define array of request body.
$request_body = array();

try {
    $response = $client->request('DELETE','https://o2r.uni-muenster.de/api/v1/compendium/{compendium_id}', ar
        'headers' => $headers,
        'json' => $request_body,
       )
    );
    print_r($response->getBody()->getContents());
 }
 catch (\GuzzleHttp\Exception\BadResponseException $e) {
    // handle exception or api errors.
    print_r($e->getMessage());
 }
```

```
 // ...
```

```
URL obj = new URL("https://o2r.uni-muenster.de/api/v1/compendium/{compendium_id}");
HttpURLConnection con = (HttpURLConnection) obj.openConnection();
con.setRequestMethod("DELETE");
int responseCode = con.getResponseCode();
BufferedReader in = new BufferedReader(
    new InputStreamReader(con.getInputStream()));
String inputLine;
StringBuffer response = new StringBuffer();
while ((inputLine = in.readLine()) != null) {
    response.append(inputLine);
}
in.close();
System.out.println(response.toString());
```

```
package main

import (
       "bytes"
       "net/http"
)

func main() {

    headers := map[string][]string{
        "Accept": []string{"application/json"},
    }

    data := bytes.NewBuffer([]byte{jsonReq})
    req, err := http.NewRequest("DELETE", "https://o2r.uni-muenster.de/api/v1/compendium/{compendium_id}", da
    req.Header = headers

    client := &http.Client{}
    resp, err := client.Do(req)
    // ...
}
```

`DELETE /compendium/{compendium_id}`

Delete compendium

To delete a compendium **candidate**, an HTTP `DELETE` request can be send to the compendium endpoint.

⚠ Important

Once a compendium is not a candidate anymore, it can only be deleted by admins with the required user level. The compendium contents should be moved to a specific location to the server as a backup.

⚠ Required user level for candidate deletion

The user deleting a candidate must be the author or have the required user level.

Request

The following request deletes the compendium with the identifier 12345, including metadata and files.

```
curl -X DELETE https://…/api/v1/compendium/12345 \
--cookie "connect.sid=<code string here>"
```

Parameters

Name	In	Type	Required	Description
compendium_id	path	string	true	Id of the compendium

> Example responses

> Unauthorized due to user not being logged in.

```
{
  "error": "user is not authenticated"
}
```

> Returns an error

```
{
  "error": "user level not sufficient to delete compendium"
}
```

```
{
  "error": "compendium not found"
}
```

Responses

Status	Meaning	Description	Schema
204	No Content	The response has an HTTP status of `204` and an empty body for successful deletion.	None
401	Unauthorized	Unauthorized due to user not being logged in.	general_error_model
403	Forbidden	Forbidden due to userlevel not being high enough.	general_error_model
404	Not Found	No compendium with the given id found.	general_error_model

To perform this operation, you must be authenticated by means of one of the following methods: cookie_authenification

download_compendium

> Code samples

```
# You can also use wget
curl -X GET https://o2r.uni-muenster.de/api/v1/compendium/{compendium_id}.{file_extension} \
  -H 'Accept: application/zip'
```

```
GET https://o2r.uni-muenster.de/api/v1/compendium/{compendium_id}.{file_extension} HTTP/1.1
Host: o2r.uni-muenster.de
```

```
Accept: application/zip
```

```
const headers = {
  'Accept':'application/zip'
};

fetch('https://o2r.uni-muenster.de/api/v1/compendium/{compendium_id}.{file_extension}',
{
  method: 'GET',

  headers: headers
})
.then(function(res) {
    return res.json();
}).then(function(body) {
    console.log(body);
});
```

```
require 'rest-client'
require 'json'

headers = {
  'Accept' => 'application/zip'
}

result = RestClient.get 'https://o2r.uni-muenster.de/api/v1/compendium/{compendium_id}.{file_extension}',
  params: {
  }, headers: headers

p JSON.parse(result)
```

```
import requests
headers = {
  'Accept': 'application/zip'
}

r = requests.get('https://o2r.uni-muenster.de/api/v1/compendium/{compendium_id}.{file_extension}', headers =

print(r.json())
```

```
<?php

require 'vendor/autoload.php';

$headers = array(
    'Accept' => 'application/zip',
);

$client = new \GuzzleHttp\Client();

// Define array of request body.
$request_body = array();

try {
    $response = $client->request('GET','https://o2r.uni-muenster.de/api/v1/compendium/{compendium_id}.{file_e
        'headers' => $headers,
        'json' => $request_body,
       )
    );
    print_r($response->getBody()->getContents());
 }
 catch (\GuzzleHttp\Exception\BadResponseException $e) {
    // handle exception or api errors.
    print_r($e->getMessage());
```

```
 }

 // ...
```

```
URL obj = new URL("https://o2r.uni-muenster.de/api/v1/compendium/{compendium_id}.{file_extension}");
HttpURLConnection con = (HttpURLConnection) obj.openConnection();
con.setRequestMethod("GET");
int responseCode = con.getResponseCode();
BufferedReader in = new BufferedReader(
    new InputStreamReader(con.getInputStream()));
String inputLine;
StringBuffer response = new StringBuffer();
while ((inputLine = in.readLine()) != null) {
    response.append(inputLine);
}
in.close();
System.out.println(response.toString());
```

```
package main

import (
       "bytes"
       "net/http"
)

func main() {

    headers := map[string][]string{
        "Accept": []string{"application/zip"},
    }

    data := bytes.NewBuffer([]byte{jsonReq})
    req, err := http.NewRequest("GET", "https://o2r.uni-muenster.de/api/v1/compendium/{compendium_id}.{file_e
    req.Header = headers

    client := &http.Client{}
    resp, err := client.Do(req)
    // ...
}
```

`GET /compendium/{compendium_id}.{file_extension}`

Download compendium

Download compendium files as an archive.

⚠ Warning

This download feature does *not* provide access to complete and valid compendia, because it does not comprise an update of the packaging, while it does include brokered metadata files. To download a valid compendium, create a shipment with the appropriate recipient.

Supported formats are as follows:

- `zip`
- `tar`
- `tar.gz`

Request

`GET /api/v1/compendium/$ID.zip`

```
GET /api/v1/compendium/:id.zip
GET /api/v1/compendium/:id.tar
GET /api/v1/compendium/:id.tar.gz
GET /api/v1/compendium/:id.tar?gzip
GET /api/v1/compendium/:id.zip?image=false
```

Parameters

Name	In	Type	Required	Description
gzip	query	string	false	*only for .tar endpoint* - compress tarball with gzip
image	query	boolean	false	`?image=true` or `?image=false` - include tarball of Docker image in the archive , default is true
compendium_id	path	string	true	Id of the compendium
file_extension	path	string	true	File Extension for Compendium Download

Detailed descriptions

gzip: *only for .tar endpoint* - compress tarball with gzip

image: `?image=true` or `?image=false` - include tarball of Docker image in the archive , default is true

Enumerated Values

Parameter	Value
file_extension	zip
file_extension	tar
file_extension	tar.gz

> Example responses

> A successfull download of a compendium as a `.zip` file.

The zip file contains a comment with the original URL.

```
$ unzip -z CXE1c.zip
Archive:  CXE1c.zip
Created by o2r [https://…/api/v1/compendium/CXE1c.zip]
```

> Returns an error

```
{
  "error": "no job found for this compendium, run a job before downloading with image"
}
```

```
{
  "error": "no compendium with this id"
}
```

Responses

Status	Meaning	Description	Schema
200	OK	The response is a file attachment. The suggested file name is available in the HTTP header content-disposition using the respective file extension for a file named with the compendium identifier (e.g. `wdpV9.zip`, `Uh1o0.tar`, or `LBIt1.tar.gz`).	None
400	Bad Request	A bad request.	general_error_model
404	Not Found	No compendium with the given id found.	general_error_model

Response Schema

This operation does not require authentication

upload_link

> Code samples

```
# You can also use wget
curl -X PUT https://o2r.uni-muenster.de/api/v1/compendium/{compendium_id}/link \
  -H 'Accept: application/json'
```

```
PUT https://o2r.uni-muenster.de/api/v1/compendium/{compendium_id}/link HTTP/1.1
Host: o2r.uni-muenster.de
Accept: application/json
```

```
const headers = {
  'Accept':'application/json'
};

fetch('https://o2r.uni-muenster.de/api/v1/compendium/{compendium_id}/link',
{
  method: 'PUT',

  headers: headers
})
.then(function(res) {
    return res.json();
}).then(function(body) {
    console.log(body);
});
```

```
require 'rest-client'
require 'json'
```

```
headers = {
  'Accept' => 'application/json'
}

result = RestClient.put 'https://o2r.uni-muenster.de/api/v1/compendium/{compendium_id}/link',
  params: {
  }, headers: headers

p JSON.parse(result)
```

```
import requests
headers = {
  'Accept': 'application/json'
}

r = requests.put('https://o2r.uni-muenster.de/api/v1/compendium/{compendium_id}/link', headers = headers)

print(r.json())
```

```
<?php

require 'vendor/autoload.php';

$headers = array(
    'Accept' => 'application/json',
);

$client = new \GuzzleHttp\Client();

// Define array of request body.
$request_body = array();

try {
    $response = $client->request('PUT','https://o2r.uni-muenster.de/api/v1/compendium/{compendium_id}/link', 
        'headers' => $headers,
        'json' => $request_body,
       )
    );
    print_r($response->getBody()->getContents());
 }
 catch (\GuzzleHttp\Exception\BadResponseException $e) {
    // handle exception or api errors.
    print_r($e->getMessage());
 }

 // ...
```

```
URL obj = new URL("https://o2r.uni-muenster.de/api/v1/compendium/{compendium_id}/link");
HttpURLConnection con = (HttpURLConnection) obj.openConnection();
con.setRequestMethod("PUT");
int responseCode = con.getResponseCode();
BufferedReader in = new BufferedReader(
    new InputStreamReader(con.getInputStream()));
String inputLine;
StringBuffer response = new StringBuffer();
while ((inputLine = in.readLine()) != null) {
    response.append(inputLine);
}
in.close();
System.out.println(response.toString());
```

```
package main
```

```
import (
       "bytes"
       "net/http"
)

func main() {

    headers := map[string][]string{
        "Accept": []string{"application/json"},
    }

    data := bytes.NewBuffer([]byte{jsonReq})
    req, err := http.NewRequest("PUT", "https://o2r.uni-muenster.de/api/v1/compendium/{compendium_id}/link", 
    req.Header = headers

    client := &http.Client{}
    resp, err := client.Do(req)
    // ...
}
```

`PUT /compendium/{compendium_id}/link`

Create link

The following request creates a link for the candidate compendium with the identifier 12345.

```
curl -X PUT https://…/api/v1/compendium/12345/link \
    --cookie "connect.sid=<code string here>"
```

Parameters

Name	In	Type	Required	Description
compendium_id	path	string	true	Id of the compendium

> Example responses

> The response has an HTTP status of `200` if the public link was created.
> The response body provides the public link `id`, which can be used for subsequent requests to:

- view the candidate compendium
- download candidate compendium files
- start jobs for the candidate compendium

Only **1 public link** is created per candidate compendium.
Subsequent requests will return the same link.


```
{
  "id": "lSaOCqxmNEO8Og42a0ONVRYUVoDWeBLr",
  "compendium_id": "xkjzY",
  "user": "0000-0002-1701-2564"
}
```


> Returns an error, due to not being authorized


```
{
  "error": "not authorized"
}
```

> Returns an error

```
{
  "error": "compendium not found"
}
```

Responses

Status	Meaning	Description	Schema
200	OK	Link creation response	link_response
401	Unauthorized	Bad Request due to not supported content_type	general_error_model
404	Not Found	Unauthorized due to user not being logged in.	general_error_model

To perform this operation, you must be authenticated by means of one of the following methods: cookie_authenification

delete_link

> Code samples

```
# You can also use wget
curl -X DELETE https://o2r.uni-muenster.de/api/v1/compendium/{compendium_id}/link \
  -H 'Accept: application/json'
```

```
DELETE https://o2r.uni-muenster.de/api/v1/compendium/{compendium_id}/link HTTP/1.1
Host: o2r.uni-muenster.de
Accept: application/json
```

```
const headers = {
  'Accept':'application/json'
};

fetch('https://o2r.uni-muenster.de/api/v1/compendium/{compendium_id}/link',
{
  method: 'DELETE',

  headers: headers
})
.then(function(res) {
    return res.json();
}).then(function(body) {
    console.log(body);
});
```

```
require 'rest-client'
require 'json'

headers = {
  'Accept' => 'application/json'
}
```

```
result = RestClient.delete 'https://o2r.uni-muenster.de/api/v1/compendium/{compendium_id}/link',
  params: {
  }, headers: headers

p JSON.parse(result)
```

```
import requests
headers = {
  'Accept': 'application/json'
}

r = requests.delete('https://o2r.uni-muenster.de/api/v1/compendium/{compendium_id}/link', headers = headers)

print(r.json())
```

```
<?php

require 'vendor/autoload.php';

$headers = array(
    'Accept' => 'application/json',
);

$client = new \GuzzleHttp\Client();

// Define array of request body.
$request_body = array();

try {
    $response = $client->request('DELETE','https://o2r.uni-muenster.de/api/v1/compendium/{compendium_id}/link
        'headers' => $headers,
        'json' => $request_body,
       )
    );
    print_r($response->getBody()->getContents());
 }
 catch (\GuzzleHttp\Exception\BadResponseException $e) {
    // handle exception or api errors.
    print_r($e->getMessage());
 }

 // ...
```

```
URL obj = new URL("https://o2r.uni-muenster.de/api/v1/compendium/{compendium_id}/link");
HttpURLConnection con = (HttpURLConnection) obj.openConnection();
con.setRequestMethod("DELETE");
int responseCode = con.getResponseCode();
BufferedReader in = new BufferedReader(
    new InputStreamReader(con.getInputStream()));
String inputLine;
StringBuffer response = new StringBuffer();
while ((inputLine = in.readLine()) != null) {
    response.append(inputLine);
}
in.close();
System.out.println(response.toString());
```

```
package main

import (
       "bytes"
       "net/http"
)
```

```
func main() {

    headers := map[string][]string{
        "Accept": []string{"application/json"},
    }

    data := bytes.NewBuffer([]byte{jsonReq})
    req, err := http.NewRequest("DELETE", "https://o2r.uni-muenster.de/api/v1/compendium/{compendium_id}/link
    req.Header = headers

    client := &http.Client{}
    resp, err := client.Do(req)
    // ...
}
```

`DELETE /compendium/{compendium_id}/link`

Create link

The following request deletes a link for the candidate compendium with the identifier 12345.

```
curl -X DELETE https://…/api/v1/compendium/12345/link \
    --cookie "connect.sid=<code string here>"
```

Parameters

Name	In	Type	Required	Description
compendium_id	path	string	true	Id of the compendium

> Example responses

> Returns an error, due to not being authorized

```
{
  "error": "not authorized"
}
```

> Returns an error

```
{
  "error": "user level not sufficient to delete compendium"
}
```

```
{
  "error": "compendium not found"
}
```

Responses

Status	Meaning	Description	Schema
204	No Content	Link deletion successfull	None

Status	Meaning	Description	Schema
401	Unauthorized	Bad Request due to not supported content_type	general_error_model
403	Forbidden	Forbidden due to user level not allowing upload	general_error_model
404	Not Found	Unauthorized due to user not being logged in.	general_error_model

To perform this operation, you must be authenticated by means of one of the following methods: cookie_authenification

get__shipment_{shipment_id}_publishment

> Code samples

```
# You can also use wget
curl -X GET https://o2r.uni-muenster.de/api/v1/shipment/{shipment_id}/publishment \
  -H 'Accept: application/json'
```

```
GET https://o2r.uni-muenster.de/api/v1/shipment/{shipment_id}/publishment HTTP/1.1
Host: o2r.uni-muenster.de
Accept: application/json
```

```
const headers = {
  'Accept':'application/json'
};

fetch('https://o2r.uni-muenster.de/api/v1/shipment/{shipment_id}/publishment',
{
  method: 'GET',

  headers: headers
})
.then(function(res) {
    return res.json();
}).then(function(body) {
    console.log(body);
});
```

```
require 'rest-client'
require 'json'

headers = {
  'Accept' => 'application/json'
}

result = RestClient.get 'https://o2r.uni-muenster.de/api/v1/shipment/{shipment_id}/publishment',
  params: {
  }, headers: headers

p JSON.parse(result)
```

```
import requests
headers = {
  'Accept': 'application/json'
}

r = requests.get('https://o2r.uni-muenster.de/api/v1/shipment/{shipment_id}/publishment', headers = headers)
```

```
print(r.json())
```

```
<?php

require 'vendor/autoload.php';

$headers = array(
    'Accept' => 'application/json',
);

$client = new \GuzzleHttp\Client();

// Define array of request body.
$request_body = array();

try {
    $response = $client->request('GET','https://o2r.uni-muenster.de/api/v1/shipment/{shipment_id}/publishment
        'headers' => $headers,
        'json' => $request_body,
       )
    );
    print_r($response->getBody()->getContents());
 }
 catch (\GuzzleHttp\Exception\BadResponseException $e) {
    // handle exception or api errors.
    print_r($e->getMessage());
 }

 // ...
```

```
URL obj = new URL("https://o2r.uni-muenster.de/api/v1/shipment/{shipment_id}/publishment");
HttpURLConnection con = (HttpURLConnection) obj.openConnection();
con.setRequestMethod("GET");
int responseCode = con.getResponseCode();
BufferedReader in = new BufferedReader(
    new InputStreamReader(con.getInputStream()));
String inputLine;
StringBuffer response = new StringBuffer();
while ((inputLine = in.readLine()) != null) {
    response.append(inputLine);
}
in.close();
System.out.println(response.toString());
```

```
package main

import (
       "bytes"
       "net/http"
)

func main() {

    headers := map[string][]string{
        "Accept": []string{"application/json"},
    }

    data := bytes.NewBuffer([]byte{jsonReq})
    req, err := http.NewRequest("GET", "https://o2r.uni-muenster.de/api/v1/shipment/{shipment_id}/publishment
    req.Header = headers

    client := &http.Client{}
    resp, err := client.Do(req)
    // ...
}
```

`GET /shipment/{shipment_id}/publishment`

List deposition files

You can request a list of all files in a deposition and their properties with the sub-resource `/publishment`.

`GET api/v1/shipment/<shipment_id>/publishment`

You can find the `id` of the file you want to interact with in this json list object at `files[n].id`, where `n` is the position of that file in the array.
Files can be identified in this response by either their id in the depot, their filename or their checksum.

Parameters

Name	In	Type	Required	Description
shipment_id	path	string	true	Id of the shipment

> Example responses

> Returns a JSON object with all the deposition files of this shipment.


```
{
  "files": [
    {
      "filesize": 393320,
      "id": "bae2a60c-bd59-47e1-a443-b34bb7d0a981",
      "filename": "4XgD9.zip",
      "checksum": "702f4db3e53b22176d1d5ddcda462a27",
      "links": {
        "self": "https://sandbox.zenodo.org/api/deposit/depositions/71552/files/bae2a60c-bd59-47e1-a443-b34bb
        "download": "https://sandbox.zenodo.org/api/files/31dc8f3d-df00-4d8a-bd99-64ef341372b3/4XgD9.zip"
      }
    }
  ]
}
```


Responses

Status	Meaning	Description	Schema
200	OK	Returns a JSON object with all the deposition files of this shipment.	shipment_files_response

This operation does not require authentication

delete__shipment_{shipment_id}*files*{file_id}

> Code samples

```
# You can also use wget
curl -X DELETE https://o2r.uni-muenster.de/api/v1/shipment/{shipment_id}/files/{file_id}
```

```
DELETE https://o2r.uni-muenster.de/api/v1/shipment/{shipment_id}/files/{file_id} HTTP/1.1
Host: o2r.uni-muenster.de
```

```
fetch('https://o2r.uni-muenster.de/api/v1/shipment/{shipment_id}/files/{file_id}',
{
  method: 'DELETE'

})
.then(function(res) {
    return res.json();
}).then(function(body) {
    console.log(body);
});
```

```
require 'rest-client'
require 'json'

result = RestClient.delete 'https://o2r.uni-muenster.de/api/v1/shipment/{shipment_id}/files/{file_id}',
  params: {
  }

p JSON.parse(result)
```

```
import requests

r = requests.delete('https://o2r.uni-muenster.de/api/v1/shipment/{shipment_id}/files/{file_id}')

print(r.json())
```

```
<?php

require 'vendor/autoload.php';

$client = new \GuzzleHttp\Client();

// Define array of request body.
$request_body = array();

try {
    $response = $client->request('DELETE','https://o2r.uni-muenster.de/api/v1/shipment/{shipment_id}/files/{f
        'headers' => $headers,
        'json' => $request_body,
       )
    );
    print_r($response->getBody()->getContents());
 }
 catch (\GuzzleHttp\Exception\BadResponseException $e) {
    // handle exception or api errors.
    print_r($e->getMessage());
 }

 // ...
```

```
URL obj = new URL("https://o2r.uni-muenster.de/api/v1/shipment/{shipment_id}/files/{file_id}");
HttpURLConnection con = (HttpURLConnection) obj.openConnection();
con.setRequestMethod("DELETE");
int responseCode = con.getResponseCode();
BufferedReader in = new BufferedReader(
    new InputStreamReader(con.getInputStream()));
String inputLine;
```

```
StringBuffer response = new StringBuffer();
while ((inputLine = in.readLine()) != null) {
    response.append(inputLine);
}
in.close();
System.out.println(response.toString());
```

```
package main

import (
       "bytes"
       "net/http"
)

func main() {

    data := bytes.NewBuffer([]byte{jsonReq})
    req, err := http.NewRequest("DELETE", "https://o2r.uni-muenster.de/api/v1/shipment/{shipment_id}/files/{f
    req.Header = headers

    client := &http.Client{}
    resp, err := client.Do(req)
    // ...
}
```

`DELETE /shipment/{shipment_id}/files/{file_id}`

Delete a specific file from a deposition

You can delete files from a `shipped` shipment's deposition. You must state a file's identifier, which can be retrieved from the shipment's deposition files property `id`, as the `file_id` path parameter. Files for a `published` shipment usually cannot be deleted.

`DELETE api/v1/shipment/<shipment_id>/files/<file_id>`

Parameters

Name	In	Type	Required	Description
shipment_id	path	string	true	Id of the shipment
file_id	path	string	true	Id of the File

Responses

Status	Meaning	Description	Schema
204	No Content	Deletion successfull	None

This operation does not require authentication

upload_substitution

> Code samples

```
# You can also use wget
curl -X POST https://o2r.uni-muenster.de/api/v1/substitution \
```

```
  -H 'Content-Type: multipart/form-data' \
  -H 'Accept: application/json'
```

```
POST https://o2r.uni-muenster.de/api/v1/substitution HTTP/1.1
Host: o2r.uni-muenster.de
Content-Type: multipart/form-data
Accept: application/json
```

```
const inputBody = '{
  "base": "G92NL",
  "overlay": "9fCTR",
  "substitutionFiles": [
    {
      "base": "climate-timeseries.csv",
      "overlay": "mytimeseries_data.csv"
    }
  ],
  "metadataHandling": "keepBase"
}';
const headers = {
  'Content-Type':'multipart/form-data',
  'Accept':'application/json'
};

fetch('https://o2r.uni-muenster.de/api/v1/substitution',
{
  method: 'POST',
  body: inputBody,
  headers: headers
})
.then(function(res) {
    return res.json();
}).then(function(body) {
    console.log(body);
});
```

```
require 'rest-client'
require 'json'

headers = {
  'Content-Type' => 'multipart/form-data',
  'Accept' => 'application/json'
}

result = RestClient.post 'https://o2r.uni-muenster.de/api/v1/substitution',
  params: {
  }, headers: headers

p JSON.parse(result)
```

```
import requests
headers = {
  'Content-Type': 'multipart/form-data',
  'Accept': 'application/json'
}

r = requests.post('https://o2r.uni-muenster.de/api/v1/substitution', headers = headers)

print(r.json())
```

```php
<?php

require 'vendor/autoload.php';

$headers = array(
    'Content-Type' => 'multipart/form-data',
    'Accept' => 'application/json',
);

$client = new \GuzzleHttp\Client();

// Define array of request body.
$request_body = array();

try {
    $response = $client->request('POST','https://o2r.uni-muenster.de/api/v1/substitution', array(
        'headers' => $headers,
        'json' => $request_body,
       )
    );
    print_r($response->getBody()->getContents());
 }
 catch (\GuzzleHttp\Exception\BadResponseException $e) {
    // handle exception or api errors.
    print_r($e->getMessage());
 }

 // ...
```

```java
URL obj = new URL("https://o2r.uni-muenster.de/api/v1/substitution");
HttpURLConnection con = (HttpURLConnection) obj.openConnection();
con.setRequestMethod("POST");
int responseCode = con.getResponseCode();
BufferedReader in = new BufferedReader(
    new InputStreamReader(con.getInputStream()));
String inputLine;
StringBuffer response = new StringBuffer();
while ((inputLine = in.readLine()) != null) {
    response.append(inputLine);
}
in.close();
System.out.println(response.toString());
```

```go
package main

import (
       "bytes"
       "net/http"
)

func main() {

    headers := map[string][]string{
        "Content-Type": []string{"multipart/form-data"},
        "Accept": []string{"application/json"},
    }

    data := bytes.NewBuffer([]byte{jsonReq})
    req, err := http.NewRequest("POST", "https://o2r.uni-muenster.de/api/v1/substitution", data)
    req.Header = headers

    client := &http.Client{}
    resp, err := client.Do(req)
    // ...
}
```

`POST /substitution`

Create substitution

`Create substitution` produces a new compendium with its own files in the storage and metadata in the database.
A substitution can be created with an HTTP `POST` request using `multipart/form-data` and content-type `JSON`.
Required content of the request are the identifiers of the base and overlay compendia and at least one pair of *substitution files*, consisting of a base file and an overlay file.

⚠ Note

A substitution process removes potentially existing packaging information, i.e. if the base compendium was a BagIt bag, the substitution will only contain the payload directory contents (`/data` directory). The overlay file is stripped of all paths and is copied directly into the substitution's root directory.

`POST /api/v1/substitution`

⚠ Required user level

The user creating a new substitution must have the required user level.

You can view the newly suvbstituted Compendium with the standard get Request for compendium view `GET /api/v1/compendium/:id`. The response will include extra metadata on the substitution.

> Body parameter

```
base: G92NL
overlay: 9fCTR
substitutionFiles:
  - base: climate-timeseries.csv
    overlay: mytimeseries_data.csv
metadataHandling: keepBase
```

Parameters

Name	In	Type	Required	Description
body	body	new_substitution	true	The Compendium to be uploaded.

Detailed descriptions

body: The Compendium to be uploaded.

> Example responses

> Creation response

```
{
  "id": "oMMFn"
}
```

> Unauthorized due to user level not being sufficient.

```
{
  "error": "not authenticated"
```

```
}
```

```
{
  "error": "not allowed"
}
```

> Not found

```
{
  "error": "base compendium not found"
}
```

```
{
  "error": "overlay compendium not found"
}
```

Responses

Status	Meaning	Description	Schema
201	Created	Creation response	upload_response
401	Unauthorized	Unauthorized due to user level not being sufficient.	general_error_model
404	Not Found	Not found	general_error_model

To perform this operation, you must be authenticated by means of one of the following methods: cookie_authenification

view_substitution

> Code samples

```
# You can also use wget
curl -X GET https://o2r.uni-muenster.de/api/v1/substitution \
  -H 'Accept: application/json'
```

```
GET https://o2r.uni-muenster.de/api/v1/substitution HTTP/1.1
Host: o2r.uni-muenster.de
Accept: application/json
```

```
const headers = {
  'Accept':'application/json'
};

fetch('https://o2r.uni-muenster.de/api/v1/substitution',
{
  method: 'GET',

  headers: headers
})
.then(function(res) {
```

```
    return res.json();
}).then(function(body) {
    console.log(body);
});
```

```
require 'rest-client'
require 'json'

headers = {
  'Accept' => 'application/json'
}

result = RestClient.get 'https://o2r.uni-muenster.de/api/v1/substitution',
  params: {
  }, headers: headers

p JSON.parse(result)
```

```
import requests
headers = {
  'Accept': 'application/json'
}

r = requests.get('https://o2r.uni-muenster.de/api/v1/substitution', headers = headers)

print(r.json())
```

```
<?php

require 'vendor/autoload.php';

$headers = array(
    'Accept' => 'application/json',
);

$client = new \GuzzleHttp\Client();

// Define array of request body.
$request_body = array();

try {
    $response = $client->request('GET','https://o2r.uni-muenster.de/api/v1/substitution', array(
        'headers' => $headers,
        'json' => $request_body,
       )
    );
    print_r($response->getBody()->getContents());
 }
 catch (\GuzzleHttp\Exception\BadResponseException $e) {
    // handle exception or api errors.
    print_r($e->getMessage());
 }

 // ...
```

```
URL obj = new URL("https://o2r.uni-muenster.de/api/v1/substitution");
HttpURLConnection con = (HttpURLConnection) obj.openConnection();
con.setRequestMethod("GET");
int responseCode = con.getResponseCode();
BufferedReader in = new BufferedReader(
    new InputStreamReader(con.getInputStream()));
String inputLine;
StringBuffer response = new StringBuffer();
while ((inputLine = in.readLine()) != null) {
```

```
    response.append(inputLine);
}
in.close();
System.out.println(response.toString());
```

```
package main

import (
       "bytes"
       "net/http"
)

func main() {

    headers := map[string][]string{
        "Accept": []string{"application/json"},
    }

    data := bytes.NewBuffer([]byte{jsonReq})
    req, err := http.NewRequest("GET", "https://o2r.uni-muenster.de/api/v1/substitution", data)
    req.Header = headers

    client := &http.Client{}
    resp, err := client.Do(req)
    // ...
}
```

`GET /substitution`

List substituted Compenium

`curl https://.../api/v1/substitution`

`GET /api/v1/substitution`

Parameters

Name	In	Type	Required	Description
overlay	query	string	false	id of the overlay compendium that the results should be related to
base	query	string	false	id of the base compendium that the results should be related to

> Example responses

> The result is a list of compendia ids which were created by a substitution process.

```
{
  "results": [
    "oMMFn",
    "asdi5",
    "nb2sg",
    "…"
  ]
}
```

> If there are no substitutions yet, the returned list is empty.

```
{
  "results": []
}
```

```
curl https://.../api/v1/substitution?base=jfL3w
```

`GET /api/v1/substitution?base=jfL3w`

Result is a list of substituted compendia based on the given base compendium.

```
{
  "results": [
    "wGmFn",
    "…"
  ]
}
```

```
curl https://.../api/v1/substitution?overlay=as4Kj
```

`GET /api/v1/substitution?overlay=as4Kj`

Result is a list of substituted compendia based on the given overlay compendium.

```
{
  "results": [
    "9pQ34",
    "1Tnd3",
    "…"
  ]
}
```

```
curl https://.../api/v1/substitution?base=lO3Td&overlay=as4Kj
```

`GET /api/v1/substitution?base=lO3Td&overlay=as4Kj`

Result is a list of substituted compendia based on the given base and overlay compendium.

```
{
  "results": [
    "9pQ34",
    "1Tnd3",
    "…"
  ]
}
```

> Not found

```
{
  "error": "base compendium undefined"
}
```

```
{
  "error": "Overlay compendium undefined"
}
```

> The successfull Listing of all the substitutions as JSON object

```
{
  "error": "not authenticated"
}
```

```
{
  "error": "not allowed"
}
```

> Not found

```
{
  "error": "base compendium not found"
}
```

```
{
  "error": "overlay compendium not found"
}
```

Responses

Status	Meaning	Description	Schema
200	OK	The successfull Listing of all the substitutions as JSON object	list_response
400	Bad Request	Not found	general_error_model
401	Unauthorized	The successfull Listing of all the substitutions as JSON object	general_error_model
404	Not Found	Not found	general_error_model

To perform this operation, you must be authenticated by means of one of the following methods: cookie_authenification

list_link

> Code samples

```
# You can also use wget
curl -X GET https://o2r.uni-muenster.de/api/v1/link \
  -H 'Accept: application/json'
```

```
GET https://o2r.uni-muenster.de/api/v1/link HTTP/1.1
Host: o2r.uni-muenster.de
Accept: application/json
```

```
const headers = {
  'Accept':'application/json'
};

fetch('https://o2r.uni-muenster.de/api/v1/link',
{
  method: 'GET',

  headers: headers
})
.then(function(res) {
    return res.json();
}).then(function(body) {
    console.log(body);
});
```

```
require 'rest-client'
require 'json'

headers = {
  'Accept' => 'application/json'
}

result = RestClient.get 'https://o2r.uni-muenster.de/api/v1/link',
  params: {
  }, headers: headers

p JSON.parse(result)
```

```
import requests
headers = {
  'Accept': 'application/json'
}

r = requests.get('https://o2r.uni-muenster.de/api/v1/link', headers = headers)

print(r.json())
```

```
<?php

require 'vendor/autoload.php';

$headers = array(
    'Accept' => 'application/json',
);

$client = new \GuzzleHttp\Client();

// Define array of request body.
$request_body = array();

try {
    $response = $client->request('GET','https://o2r.uni-muenster.de/api/v1/link', array(
        'headers' => $headers,
        'json' => $request_body,
       )
    );
    print_r($response->getBody()->getContents());
 }
 catch (\GuzzleHttp\Exception\BadResponseException $e) {
    // handle exception or api errors.
    print_r($e->getMessage());
 }

 // ...
```

```
URL obj = new URL("https://o2r.uni-muenster.de/api/v1/link");
HttpURLConnection con = (HttpURLConnection) obj.openConnection();
con.setRequestMethod("GET");
int responseCode = con.getResponseCode();
BufferedReader in = new BufferedReader(
    new InputStreamReader(con.getInputStream()));
String inputLine;
StringBuffer response = new StringBuffer();
while ((inputLine = in.readLine()) != null) {
    response.append(inputLine);
}
in.close();
System.out.println(response.toString());
```

```
package main

import (
       "bytes"
       "net/http"
)

func main() {

    headers := map[string][]string{
        "Accept": []string{"application/json"},
    }

    data := bytes.NewBuffer([]byte{jsonReq})
    req, err := http.NewRequest("GET", "https://o2r.uni-muenster.de/api/v1/link", data)
    req.Header = headers

    client := &http.Client{}
    resp, err := client.Do(req)
    // ...
}
```

`GET /link`

List links

```
curl -X GET http://localhost/api/v1/link
     --cookie "connect.sid=s%3AVU61x4E<rest of cookie>"
```

> Example responses

> Link listing response

```
{
  "results": [
    {
      "id": "b56Cy5EG7oiCBPCZMjXIPXoSyPVxiGVA",
      "compendium_id": "eENFZ",
      "user": "0000-0002-1701-2564"
    },
    {
      "id": "p6s3GGn6EaDoZXM8jOWuNd5E1lHKPVRt",
      "compendium_id": "xkjzY",
      "user": "0000-0002-1701-2564"
    }
  ]
}
```

> Returns an error, due to not being authorized

```
{
  "error": "not authorized"
}
```

Responses

Status	Meaning	Description	Schema
200	OK	Link listing response	link_list_response
401	Unauthorized	Bad Request due to not supported content_type	general_error_model

To perform this operation, you must be authenticated by means of one of the following methods: cookie_authenification

simple_search

> Code samples

```
# You can also use wget
curl -X GET https://o2r.uni-muenster.de/api/v1/search?q=string \
  -H 'Accept: application/json'
```

```
GET https://o2r.uni-muenster.de/api/v1/search?q=string HTTP/1.1
Host: o2r.uni-muenster.de
Accept: application/json
```

```
const headers = {
  'Accept':'application/json'
};

fetch('https://o2r.uni-muenster.de/api/v1/search?q=string',
{
  method: 'GET',

  headers: headers
})
.then(function(res) {
    return res.json();
}).then(function(body) {
    console.log(body);
});
```

```
require 'rest-client'
require 'json'

headers = {
  'Accept' => 'application/json'
}

result = RestClient.get 'https://o2r.uni-muenster.de/api/v1/search',
  params: {
  'q' => 'string'
```

```
}, headers: headers

p JSON.parse(result)
```

```
import requests
headers = {
  'Accept': 'application/json'
}

r = requests.get('https://o2r.uni-muenster.de/api/v1/search', params={
  'q': 'string'
}, headers = headers)

print(r.json())
```

```
<?php

require 'vendor/autoload.php';

$headers = array(
    'Accept' => 'application/json',
);

$client = new \GuzzleHttp\Client();

// Define array of request body.
$request_body = array();

try {
    $response = $client->request('GET','https://o2r.uni-muenster.de/api/v1/search', array(
        'headers' => $headers,
        'json' => $request_body,
       )
    );
    print_r($response->getBody()->getContents());
 }
 catch (\GuzzleHttp\Exception\BadResponseException $e) {
    // handle exception or api errors.
    print_r($e->getMessage());
 }

 // ...
```

```
URL obj = new URL("https://o2r.uni-muenster.de/api/v1/search?q=string");
HttpURLConnection con = (HttpURLConnection) obj.openConnection();
con.setRequestMethod("GET");
int responseCode = con.getResponseCode();
BufferedReader in = new BufferedReader(
    new InputStreamReader(con.getInputStream()));
String inputLine;
StringBuffer response = new StringBuffer();
while ((inputLine = in.readLine()) != null) {
    response.append(inputLine);
}
in.close();
System.out.println(response.toString());
```

```
package main

import (
       "bytes"
       "net/http"
)
```

```
func main() {

    headers := map[string][]string{
        "Accept": []string{"application/json"},
    }

    data := bytes.NewBuffer([]byte{jsonReq})
    req, err := http.NewRequest("GET", "https://o2r.uni-muenster.de/api/v1/search", data)
    req.Header = headers

    client := &http.Client{}
    resp, err := client.Do(req)
    // ...
}
```

`GET /search`

*🚧 Simple Search
*

🚧 This endpoint is *not* available on the public demo server. 🚧

The search uses a document database to provide high speed and powerful search capabilities for compendia, including spatial and temporal properties.

The search structure is based on Elasticsearch and thereby eases an implementation, because the requests and responses shown here can be directly mapped to respectively from Elasticsearch's API.

Indexed information:

- compendium *metadata* (including harvested and user-edited metadata such as temporal ranges and spatial extents)
- *full texts* of text files in a compendium

Simple search

A simple search allows searching for search terms using an `HTTP GET` request accepting `application/json` content type.

`curl -H 'Content-Type: application/json' https://.../api/v1/search?q=$SEARCHTERM`

Example requests

- http://o2r.uni-muenster.de/api/v1/search?q=*
- http://o2r.uni-muenster.de/api/v1/search?q=europe%20temperature%20data%20analysis
- http://o2r.uni-muenster.de/api/v1/search?q=10.5555%2F12345678
- http://o2r.uni-muenster.de/api/v1/search?q=geo&resources=compendia
- http://o2r.uni-muenster.de/api/v1/search?q=failure&resources=jobs

Parameters

Name	In	Type	Required	Description
q	query	string	true	search term(s), must be URL-encoded
resources	query	string	false	a comma-separated list of resources to include in the search;

Detailed descriptions

q: search term(s), must be URL-encoded

resources: a comma-separated list of resources to include in the search;

Enumerated Values

Parameter	Value
resources	compendia
resources	jobs
resources	all

> Example responses

⚠ Note

The available metadata is a synced clone of the compendium metadata stored in the main database. For more information on the mapping from the main database to the search database, take a look at the `o2r-finder` microservice

```
{
  "hits": {
    "total": 1,
    "max_score": 1.0586987,
    "hits": [
      {
        "_score": 1.0586987,
        "_source": {
          "metadata": {
            "o2r": "..."
          }
        }
      }
    ]
  }
}
```

Responses

Status	Meaning	Description	Schema
200	OK	The **response** is `JSON` with the root element is `hits`, which has the same as the `hits` element from an Elasticsearch response but does not include internal fields such as `_index`, `_type`, and `_id`.	simple_search_response

This operation does not require authentication

complex_search

> Code samples

```
# You can also use wget
curl -X POST https://o2r.uni-muenster.de/api/v1/search \
  -H 'Content-Type: multipart/json' \
  -H 'Accept: application/json'
```

```
POST https://o2r.uni-muenster.de/api/v1/search HTTP/1.1
Host: o2r.uni-muenster.de
Content-Type: multipart/json
Accept: application/json
```

```
const inputBody = '{
  "query": {
    "bool": {
      "must": {
        "match_all": {}
      },
      "filter": [
        {
          "range": {
            "metadata.o2r.temporal.begin": {
              "from": "2015-03-01T00:00:00.000Z"
            }
          }
        },
        {
          "range": {
            "metadata.o2r.temporal.end": {
              "to": "2017-10-01T00:00:00.000Z"
            }
          }
        }
      ]
    }
  },
  "from": 0,
  "size": 10
}';
const headers = {
  'Content-Type':'multipart/json',
  'Accept':'application/json'
};

fetch('https://o2r.uni-muenster.de/api/v1/search',
{
  method: 'POST',
  body: inputBody,
  headers: headers
})
.then(function(res) {
    return res.json();
}).then(function(body) {
    console.log(body);
});
```

```
require 'rest-client'
require 'json'

headers = {
  'Content-Type' => 'multipart/json',
  'Accept' => 'application/json'
}

result = RestClient.post 'https://o2r.uni-muenster.de/api/v1/search',
  params: {
  }, headers: headers

p JSON.parse(result)
```

```python
import requests
headers = {
  'Content-Type': 'multipart/json',
  'Accept': 'application/json'
}

r = requests.post('https://o2r.uni-muenster.de/api/v1/search', headers = headers)

print(r.json())
```

```php
<?php

require 'vendor/autoload.php';

$headers = array(
    'Content-Type' => 'multipart/json',
    'Accept' => 'application/json',
);

$client = new \GuzzleHttp\Client();

// Define array of request body.
$request_body = array();

try {
    $response = $client->request('POST','https://o2r.uni-muenster.de/api/v1/search', array(
        'headers' => $headers,
        'json' => $request_body,
       )
    );
    print_r($response->getBody()->getContents());
 }
 catch (\GuzzleHttp\Exception\BadResponseException $e) {
    // handle exception or api errors.
    print_r($e->getMessage());
 }

 // ...
```

```java
URL obj = new URL("https://o2r.uni-muenster.de/api/v1/search");
HttpURLConnection con = (HttpURLConnection) obj.openConnection();
con.setRequestMethod("POST");
int responseCode = con.getResponseCode();
BufferedReader in = new BufferedReader(
    new InputStreamReader(con.getInputStream()));
String inputLine;
StringBuffer response = new StringBuffer();
while ((inputLine = in.readLine()) != null) {
    response.append(inputLine);
}
in.close();
System.out.println(response.toString());
```

```go
package main

import (
       "bytes"
       "net/http"
)

func main() {

    headers := map[string][]string{
        "Content-Type": []string{"multipart/json"},
        "Accept": []string{"application/json"},
```

```
    }

    data := bytes.NewBuffer([]byte{jsonReq})
    req, err := http.NewRequest("POST", "https://o2r.uni-muenster.de/api/v1/search", data)
    req.Header = headers

    client := &http.Client{}
    resp, err := client.Do(req)
    // ...
}
```

`POST /search`

*🚧 Complex Search
*

Complex Search

A complex search is enabled via `POST` requests with a `JSON` payload as `HTTP POST` data (*not* `multipart/form-data`) accepting an `application/json` content type as response.
Queries can include filters, aggregation and spatio-temporal operations as defined in the Elasticsearch Query DSL.

```
curl -X POST -H 'Content-Type: application/json' 'https://.../api/v1/search' -d '$QUERY_DSL'
```

The **response** structure is the same as for simple search.

⚠ Note

Use the index names `compendia` and `jobs` in a terms query to only retrieve one resource type.

> Body parameter

> In this example a filter has been nested within a boolean/must match query.
> The filter has been applied to the `metadata.o2r.spatial.geometry` field of the dataset with a `within` relation so that only compendia with a spatial extent completely contained in the provided shape are fetched.

Parameters

Name	In	Type	Required	Description
body	body	new_search	true	The following fields are especially relevant to build queries.

Detailed descriptions

body: The following fields are especially relevant to build queries.

- `metadata.o2r.temporal.begin` and `metadata.o2r.temporal.end` provide a compendium's temporal extent
- `metadata.o2r.spatial.geometry` has the compendium's spatial extent

Besides these fields, all metadata of the `o2r` metadata format can be used.

> Example responses

⚠ Note

The available metadata is a synced clone of the compendium metadata stored in the main database. For more information on the mapping from the main database to the search database, take a look at the `o2r-finder` microservice

```
{
  "hits": {
    "total": 1,
    "max_score": 1.0586987,
    "hits": [
      {
        "_score": 1.0586987,
        "_source": {
          "metadata": {
            "o2r": "..."
          }
        }
      }
    ]
  }
}
```

Responses

Status	Meaning	Description	Schema
200	OK	The **response** is `JSON` with the root element is `hits`, which has the same as the `hits` element from an Elasticsearch response but does not include internal fields such as `_index`, `_type`, and `_id`.	simple_search_response

This operation does not require authentication

Metadata

Basics

Metadata in a compendium is stored in a directory `.erc`. This directory contains the normative metadata documents using a file naming scheme `<PREFIX>_<MODEL>_<VERSION>.<FORMAT>` filled via each metadata mapping file found in the broker tool of the o2r metadata tool suite, the default prefix is `metadata`, e.g. `metadata_o2r_1.json`, `metadata_zenodo_1.json`, or `metadata_datacite_41.xml`. The filename of the extracted raw metadata has no versioning and is constantly found as `metadata_raw.json`.

A copy of the files in this directory is kept in database for easier access, so every compendium returned by the API can contain different sub-properties in the metadata property.
This API always returns the database copy of the metadata elements.
You can download the respective files to access the normative metadata documents.

Metadata formats

The files are available on demand, but metadata variants are created after each metadata update.

The sub-properties of the `metadata` and their content are

- `raw` contains raw metadata extracted automatically
- `o2r` holds the **main information for display** and is modelled according the the o2r metadata model. This metadata is reviewed by the user and the basis for translating to other metadata formats and also for search.
- `zenodo` holds Zenodo metadata for shipments made to Zenodo and is brokered from `o2r` metadata
- `zenodo_sandbox` holds Zenodo metadata for shipments made to Zenodo Sandbox, i.e. a clone of `zenodo` metadata

⚠ Note

The information in each sub-property are subject to independent workflows and may differ from one another. The term **brokering** is used for translation from one metadata format into another.

Metadata validation

Only valid metadata can be saved to a compendium.
The `o2r` metadata element is validated against a JSON Schema using the `validate` tool of `o2r-meta`.
The schema file is included in the `o2r-meta` repository: https://raw.githubusercontent.com/o2r-project/o2r-meta/master/schema/json/o2r-meta-schema.json.

get_metadata

> Code samples

```
# You can also use wget
curl -X GET https://o2r.uni-muenster.de/api/v1/compendium/{compendium_id}/metadata \
  -H 'Accept: application/json'
```

```
GET https://o2r.uni-muenster.de/api/v1/compendium/{compendium_id}/metadata HTTP/1.1
Host: o2r.uni-muenster.de
Accept: application/json
```

```
const headers = {
  'Accept':'application/json'
};

fetch('https://o2r.uni-muenster.de/api/v1/compendium/{compendium_id}/metadata',
{
  method: 'GET',

  headers: headers
})
.then(function(res) {
    return res.json();
}).then(function(body) {
    console.log(body);
});
```

```
require 'rest-client'
require 'json'

headers = {
  'Accept' => 'application/json'
}

result = RestClient.get 'https://o2r.uni-muenster.de/api/v1/compendium/{compendium_id}/metadata',
  params: {
  }, headers: headers

p JSON.parse(result)
```

```
import requests
headers = {
  'Accept': 'application/json'
}

r = requests.get('https://o2r.uni-muenster.de/api/v1/compendium/{compendium_id}/metadata', headers = headers)
```

```
print(r.json())
```

```
<?php

require 'vendor/autoload.php';

$headers = array(
    'Accept' => 'application/json',
);

$client = new \GuzzleHttp\Client();

// Define array of request body.
$request_body = array();

try {
    $response = $client->request('GET','https://o2r.uni-muenster.de/api/v1/compendium/{compendium_id}/metadat
        'headers' => $headers,
        'json' => $request_body,
       )
    );
    print_r($response->getBody()->getContents());
 }
 catch (\GuzzleHttp\Exception\BadResponseException $e) {
    // handle exception or api errors.
    print_r($e->getMessage());
 }

 // ...
```

```
URL obj = new URL("https://o2r.uni-muenster.de/api/v1/compendium/{compendium_id}/metadata");
HttpURLConnection con = (HttpURLConnection) obj.openConnection();
con.setRequestMethod("GET");
int responseCode = con.getResponseCode();
BufferedReader in = new BufferedReader(
    new InputStreamReader(con.getInputStream()));
String inputLine;
StringBuffer response = new StringBuffer();
while ((inputLine = in.readLine()) != null) {
    response.append(inputLine);
}
in.close();
System.out.println(response.toString());
```

```
package main

import (
       "bytes"
       "net/http"
)

func main() {

    headers := map[string][]string{
        "Accept": []string{"application/json"},
    }

    data := bytes.NewBuffer([]byte{jsonReq})
    req, err := http.NewRequest("GET", "https://o2r.uni-muenster.de/api/v1/compendium/{compendium_id}/metadat
    req.Header = headers

    client := &http.Client{}
    resp, err := client.Do(req)
    // ...
}
```

`GET /compendium/{compendium_id}/metadata`

Get the compendium metadata

The following endpoint allows to access only the normative o2r-metadata element:

`curl https://…/api/v1/$ID/metadata`

`GET /api/v1/compendium/:id/metadata`

Parameters

Name	In	Type	Required	Description
compendium_id	path	string	true	Id of the compendium

> Example responses

> Returns a json object with the metadata

```
{
  "id": "compendium_id",
  "metadata": {
    "o2r": {
      "...": "..."
    }
  }
}
```

> For discovery purposes, the metadata includes extracted GeoJSON bounding boxes based on data files in a workspace.

Currently supported spatial data sources:

- shapefiles

The following structure is made available per file.

The `spatial` key has a `union` bounding box, that wraps all extracted bounding boxes.

```
{
  "spatial": {
    "files": [
      {
        "geojson": {
          "bbox": [
            -2.362060546875,
            52.0862573323384,
            -1.285400390625,
            52.649729197309426
          ],
          "geometry": {
            "coordinates": [
              [
                [
                  -2.362060546875,
                  52.0862573323384
                ],
                [
                  -1.285400390625,
```

```
                  52.649729197309426
                ]
              ]
            ],
            "type": "Polygon"
          },
          "type": "Feature"
        },
        "source_file": "path/to/file1.geojson"
      },
      {
        "geojson": {
          "bbox": [
            7.595369517803192,
            51.96245837645124,
            7.62162297964096,
            51.96966694957956
          ],
          "geometry": {
            "coordinates": [
              [
                [
                  7.595369517803192,
                  51.96245837645124
                ],
                [
                  7.62162297964096,
                  51.96966694957956
                ]
              ]
            ],
            "type": "Polygon"
          },
          "type": "Feature"
        },
        "source_file": "path/to/file2.shp"
      }
    ],
    "union": {
      "geojson": {
        "bbox": [
          -2.362060546875,
          51.96245837645124,
          7.62162297964096,
          51.96245837645124
        ],
        "geometry": {
          "coordinates": [
            [
              -2.362060546875,
              51.96245837645124
            ],
            [
              7.62162297964096,
              51.96245837645124
            ],
            [
              7.62162297964096,
              52.649729197309426
            ],
            [
              -2.362060546875,
              52.649729197309426
            ]
          ],
          "type": "Polygon"
        },
        "type": "Feature"
      }
    }
  }
}
```

> Returns an error

```
{
  "error": "no compendium with this id"
}
```

Responses

Status	Meaning	Description	Schema
200	OK	The o2r-metadata object of the specified compendium	metadata_response
404	Not Found	No compendium with the given id found.	general_error_model

To perform this operation, you must be authenticated by means of one of the following methods: cookie_authenification

update_metadata

> Code samples

```
# You can also use wget
curl -X PUT https://o2r.uni-muenster.de/api/v1/compendium/{compendium_id}/metadata \
  -H 'Accept: application/json'
```

```
PUT https://o2r.uni-muenster.de/api/v1/compendium/{compendium_id}/metadata HTTP/1.1
Host: o2r.uni-muenster.de
Accept: application/json
```

```
const headers = {
  'Accept':'application/json'
};

fetch('https://o2r.uni-muenster.de/api/v1/compendium/{compendium_id}/metadata',
{
  method: 'PUT',

  headers: headers
})
.then(function(res) {
    return res.json();
}).then(function(body) {
    console.log(body);
});
```

```
require 'rest-client'
require 'json'

headers = {
  'Accept' => 'application/json'
}

result = RestClient.put 'https://o2r.uni-muenster.de/api/v1/compendium/{compendium_id}/metadata',
  params: {
  }, headers: headers

p JSON.parse(result)
```

```
```

```
import requests
headers = {
  'Accept': 'application/json'
}

r = requests.put('https://o2r.uni-muenster.de/api/v1/compendium/{compendium_id}/metadata', headers = headers)

print(r.json())
```

```
<?php

require 'vendor/autoload.php';

$headers = array(
    'Accept' => 'application/json',
);

$client = new \GuzzleHttp\Client();

// Define array of request body.
$request_body = array();

try {
    $response = $client->request('PUT','https://o2r.uni-muenster.de/api/v1/compendium/{compendium_id}/metadat
        'headers' => $headers,
        'json' => $request_body,
       )
    );
    print_r($response->getBody()->getContents());
 }
 catch (\GuzzleHttp\Exception\BadResponseException $e) {
    // handle exception or api errors.
    print_r($e->getMessage());
 }

 // ...
```

```
URL obj = new URL("https://o2r.uni-muenster.de/api/v1/compendium/{compendium_id}/metadata");
HttpURLConnection con = (HttpURLConnection) obj.openConnection();
con.setRequestMethod("PUT");
int responseCode = con.getResponseCode();
BufferedReader in = new BufferedReader(
    new InputStreamReader(con.getInputStream()));
String inputLine;
StringBuffer response = new StringBuffer();
while ((inputLine = in.readLine()) != null) {
    response.append(inputLine);
}
in.close();
System.out.println(response.toString());
```

```
package main

import (
       "bytes"
       "net/http"
)

func main() {

    headers := map[string][]string{
        "Accept": []string{"application/json"},
    }
```

```
    data := bytes.NewBuffer([]byte{jsonReq})
    req, err := http.NewRequest("PUT", "https://o2r.uni-muenster.de/api/v1/compendium/{compendium_id}/metadat
    req.Header = headers

    client := &http.Client{}
    resp, err := client.Do(req)
    // ...
}
```

`PUT /compendium/{compendium_id}/metadata`

Update the compendium metadata

The following endpoint can be used to update the `o2r` metadata elements.
All other metadata sub-properties are only updated by the service itself, i.e. brokered metadata.
After creation the metadata is persisted to both files and database, so updating the metadata via this endpoint allows to trigger a brokering process and to retrieve different metadata formats either via this metadata API or via downloading the respective file using the download endpoint.

⚠ Metadata update rights

Only authors of a compendium or users with the required user level can update a compendium's metadata.

Metadata update request

```
curl -H 'Content-Type: application/json' \
  -X PUT \
  --cookie "connect.sid=<code string here>" \
  -d '{ "o2r": { "title": "Blue Book" } }' \
  /api/v1/compendium/:id/metadata
```

The request overwrites the existing metadata properties, so the full o2r metadata must be put with a JSON object called `o2r` at the root, even if only specific fields are changed.

Parameters

Name	In	Type	Required	Description
compendium_id	path	string	true	Id of the compendium

> Example responses

> Returns a json object with the updated excerpt.

```
{
  "id": "compendium_id",
  "metadata": {
    "o2r": {
      "title": "Blue Book"
    }
  }
}
```

> Returns an error due to missing metadata in the request (e.g. description property missing)

```
{
  "error": "Error updating metadata file, see log for details",
  "log": "[o2rmeta] 20180302.085940 received arguments: {'debug': True, 'tool': 'validate', 'schema': 'schema
}
```


> Bad Request


```
"SyntaxEroor [...]"
```


> Returns an error, due to not being authorized


```
{
  "error": "not authorized"
}
```


> The JSON doesnt meet the requirements.


```
{
  "error": "JSON with root element 'o2r' required"
}
```


Responses

Status	Meaning	Description	Schema
200	OK	Successfull metdata update	metadata_response
400	Bad Request	Bad Request	None
401	Unauthorized	Unauthorized	general_error_model
422	Unprocessable Entity	Unprocessable Enitity	general_error_model

Response Schema

To perform this operation, you must be authenticated by means of one of the following methods: cookie_authenification

Execution

Execute a compendium

Execution jobs are used to run the analysis in a compendium.
When a new execution job is started, the contents of the research compendium are cloned to create a trackable execution (see Job files).
The status information, logs and final working directory data are saved in their final state, so that they can be reviewed later on.

All execution jobs are tied to a single research compendium and reflect the execution history of that research compendium.

A trivial execution job would be a completely unmodified compendium, to test the executability and thus basic reproducibility of the contained workflow.

Job files

All files except the following are copied to a separate storage for each job:

- existing image tarballs, e.g. `image.tar` to reduce size of copied files and because jobs use images from the local image repository anyway
- the *display file* to make sure the check does not wrongly work on the original display file

Job status

The property `job.status` shows the **overall status** of a job.

The overall status can be one of following:

- `success` - if status of all steps is `success`.
- `failure` - if status of at least one step is `failure`.
- `running` - if status of at least one step is `running` and no status is `failure`.

More information about `steps` can be found in subsection `Steps` of section `View single job`.

Steps of a job

One job consists of a series of steps.
The are executed in order.

- **validate_bag**
  Validate the BagIt bag using the npm library bagit; may be skipped if compendium is not a bag, will usually fail because of added metadata files during upload.
- **generate_configuration**
  Create a compendium configuration file; may be skipped if configuration file is already present.
- **validate_compendium**
  Parses and validates the bagtainer configuration file.
- **generate_manifest**
  Executes the given analysis to create a container manifest; may be skipped if manifest file is already present.
- **image_prepare**
  Create an archive of the payload (i.e. the workspace, or the data in a BagIt bag), which allows to build and run the image also on remote hosts.
- **image_build**
  Build an image and tag it `erc:<job_id>`.
- **image_execute**
  Run the container and return based on status code of program that ran inside the container.
- **check**
  Run a check on the contents of the container. Validate the results of the executed calculations. The check provides either a list of errors or a reference to displayable content in the property `display.diff`.
- **image_save**
  Export the image to a file within the compendium directory (potentially a large file!). This is skipped if the check failed.
- **cleanup**
  Remove image or job files (depending on server-side settings).

Job status updates

You can subscribe to real time status updates on jobs using WebSockets.
The implementation is based on socket.io and using their client is recommended.

The job log is available at `https://o2r.uni-muenster.de` under the namespace `api/v1/logs/job`.

```
# create a socket.io client:
var socket = io('https://o2r.uni-muenster.de/api/v1/logs/job');
```

compendium_list_job

> Code samples

```
# You can also use wget
curl -X GET https://o2r.uni-muenster.de/api/v1/compendium/{compendium_id}/jobs \
  -H 'Accept: application/json'
```

```
GET https://o2r.uni-muenster.de/api/v1/compendium/{compendium_id}/jobs HTTP/1.1
Host: o2r.uni-muenster.de
Accept: application/json
```

```
const headers = {
  'Accept':'application/json'
};

fetch('https://o2r.uni-muenster.de/api/v1/compendium/{compendium_id}/jobs',
{
  method: 'GET',

  headers: headers
})
.then(function(res) {
    return res.json();
}).then(function(body) {
    console.log(body);
});
```

```
require 'rest-client'
require 'json'

headers = {
  'Accept' => 'application/json'
}

result = RestClient.get 'https://o2r.uni-muenster.de/api/v1/compendium/{compendium_id}/jobs',
  params: {
  }, headers: headers

p JSON.parse(result)
```

```
import requests
headers = {
  'Accept': 'application/json'
}

r = requests.get('https://o2r.uni-muenster.de/api/v1/compendium/{compendium_id}/jobs', headers = headers)
```

```
print(r.json())
```

```
<?php

require 'vendor/autoload.php';

$headers = array(
    'Accept' => 'application/json',
);

$client = new \GuzzleHttp\Client();

// Define array of request body.
$request_body = array();

try {
    $response = $client->request('GET','https://o2r.uni-muenster.de/api/v1/compendium/{compendium_id}/jobs', 
        'headers' => $headers,
        'json' => $request_body,
       )
    );
    print_r($response->getBody()->getContents());
 }
 catch (\GuzzleHttp\Exception\BadResponseException $e) {
    // handle exception or api errors.
    print_r($e->getMessage());
 }

 // ...
```

```
URL obj = new URL("https://o2r.uni-muenster.de/api/v1/compendium/{compendium_id}/jobs");
HttpURLConnection con = (HttpURLConnection) obj.openConnection();
con.setRequestMethod("GET");
int responseCode = con.getResponseCode();
BufferedReader in = new BufferedReader(
    new InputStreamReader(con.getInputStream()));
String inputLine;
StringBuffer response = new StringBuffer();
while ((inputLine = in.readLine()) != null) {
    response.append(inputLine);
}
in.close();
System.out.println(response.toString());
```

```
package main

import (
       "bytes"
       "net/http"
)

func main() {

    headers := map[string][]string{
        "Accept": []string{"application/json"},
    }

    data := bytes.NewBuffer([]byte{jsonReq})
    req, err := http.NewRequest("GET", "https://o2r.uni-muenster.de/api/v1/compendium/{compendium_id}/jobs", 
    req.Header = headers

    client := &http.Client{}
    resp, err := client.Do(req)
    // ...
}
```

`GET /compendium/{compendium_id}/jobs`

List related execution jobs

`curl https://…/api/v1/compendium/$ID/jobs`

`GET /api/v1/compendium/:id/jobs`

Parameters

Name	In	Type	Required	Description
compendium_id	path	string	true	Id of the compendium

> Example responses

> Returns a JSON object with the jobs related to the compendium

```
{
  "results": [
    "nkm4L",
    "asdi5",
    "nb2sg",
    "…"
  ]
}
```

> If there are no jobs, the returned list is empty.

```
{
  "results": []
}
```

Responses

Status	Meaning	Description	Schema
200	OK	Returns a JSON object with the jobs related to the compendium	None

Response Schema

This operation does not require authentication

create_job

> Code samples

```
# You can also use wget
curl -X POST https://o2r.uni-muenster.de/api/v1/job \
  -H 'Content-Type: multipart/form-data' \
  -H 'Accept: application/json'
```

```
POST https://o2r.uni-muenster.de/api/v1/job HTTP/1.1
Host: o2r.uni-muenster.de
Content-Type: multipart/form-data
Accept: application/json
```

```
const inputBody = '{
  "compendium_id": "string"
}';
const headers = {
  'Content-Type':'multipart/form-data',
  'Accept':'application/json'
};

fetch('https://o2r.uni-muenster.de/api/v1/job',
{
  method: 'POST',
  body: inputBody,
  headers: headers
})
.then(function(res) {
    return res.json();
}).then(function(body) {
    console.log(body);
});
```

```
require 'rest-client'
require 'json'

headers = {
  'Content-Type' => 'multipart/form-data',
  'Accept' => 'application/json'
}

result = RestClient.post 'https://o2r.uni-muenster.de/api/v1/job',
  params: {
  }, headers: headers

p JSON.parse(result)
```

```
import requests
headers = {
  'Content-Type': 'multipart/form-data',
  'Accept': 'application/json'
}

r = requests.post('https://o2r.uni-muenster.de/api/v1/job', headers = headers)

print(r.json())
```

```
<?php

require 'vendor/autoload.php';

$headers = array(
    'Content-Type' => 'multipart/form-data',
    'Accept' => 'application/json',
);

$client = new \GuzzleHttp\Client();

// Define array of request body.
$request_body = array();
```

```
try {
    $response = $client->request('POST','https://o2r.uni-muenster.de/api/v1/job', array(
        'headers' => $headers,
        'json' => $request_body,
       )
    );
    print_r($response->getBody()->getContents());
 }
 catch (\GuzzleHttp\Exception\BadResponseException $e) {
    // handle exception or api errors.
    print_r($e->getMessage());
 }

 // ...
```

```
URL obj = new URL("https://o2r.uni-muenster.de/api/v1/job");
HttpURLConnection con = (HttpURLConnection) obj.openConnection();
con.setRequestMethod("POST");
int responseCode = con.getResponseCode();
BufferedReader in = new BufferedReader(
    new InputStreamReader(con.getInputStream()));
String inputLine;
StringBuffer response = new StringBuffer();
while ((inputLine = in.readLine()) != null) {
    response.append(inputLine);
}
in.close();
System.out.println(response.toString());
```

```
package main

import (
       "bytes"
       "net/http"
)

func main() {

    headers := map[string][]string{
        "Content-Type": []string{"multipart/form-data"},
        "Accept": []string{"application/json"},
    }

    data := bytes.NewBuffer([]byte{jsonReq})
    req, err := http.NewRequest("POST", "https://o2r.uni-muenster.de/api/v1/job", data)
    req.Header = headers

    client := &http.Client{}
    resp, err := client.Do(req)
    // ...
}
```

`POST /job`

Create a new job

Create and run a new execution job with an HTTP POST request using multipart/form-data. Requires a compendium_id.

⚠ Required user level and authentication

The user creating a new compendium must have the required user level. Requests must be authenticated with a cookie `connect.sid`, see user authentication.

```
curl -F compendium_id=$ID https://…/api/v1/job
```

`POST /api/v1/job`

> Body parameter

```
compendium_id: string
```

Parameters

Name	In	Type	Required	Description
body	body	new_job	true	The Compendium to be uploaded.

Detailed descriptions

body: The Compendium to be uploaded.

> Example responses

> Returns the id of the new job

```
{
  "job_id": "ngK4m"
}
```

> Returns an error

```
{
  "error": "no compendium with this id"
}
```

```
{
  "error": "could not create job"
}
```

Responses

Status	Meaning	Description	Schema
200	OK	A successfull job creation	job_response
404	Not Found	No matching compendium found	general_error_model
500	Internal Server Error	Internal server Error	general_error_model

To perform this operation, you must be authenticated by means of one of the following methods: cookie_authenification

list_job

Code samples

```
# You can also use wget
curl -X GET https://o2r.uni-muenster.de/api/v1/job \
  -H 'Accept: application/json'
```

```
GET https://o2r.uni-muenster.de/api/v1/job HTTP/1.1
Host: o2r.uni-muenster.de
Accept: application/json
```

```
const headers = {
  'Accept':'application/json'
};

fetch('https://o2r.uni-muenster.de/api/v1/job',
{
  method: 'GET',

  headers: headers
})
.then(function(res) {
    return res.json();
}).then(function(body) {
    console.log(body);
});
```

```
require 'rest-client'
require 'json'

headers = {
  'Accept' => 'application/json'
}

result = RestClient.get 'https://o2r.uni-muenster.de/api/v1/job',
  params: {
  }, headers: headers

p JSON.parse(result)
```

```
import requests
headers = {
  'Accept': 'application/json'
}

r = requests.get('https://o2r.uni-muenster.de/api/v1/job', headers = headers)

print(r.json())
```

```
<?php

require 'vendor/autoload.php';

$headers = array(
    'Accept' => 'application/json',
);

$client = new \GuzzleHttp\Client();

// Define array of request body.
```

```
$request_body = array();

try {
    $response = $client->request('GET','https://o2r.uni-muenster.de/api/v1/job', array(
        'headers' => $headers,
        'json' => $request_body,
       )
    );
    print_r($response->getBody()->getContents());
 }
 catch (\GuzzleHttp\Exception\BadResponseException $e) {
    // handle exception or api errors.
    print_r($e->getMessage());
 }

 // ...
```

```
URL obj = new URL("https://o2r.uni-muenster.de/api/v1/job");
HttpURLConnection con = (HttpURLConnection) obj.openConnection();
con.setRequestMethod("GET");
int responseCode = con.getResponseCode();
BufferedReader in = new BufferedReader(
    new InputStreamReader(con.getInputStream()));
String inputLine;
StringBuffer response = new StringBuffer();
while ((inputLine = in.readLine()) != null) {
    response.append(inputLine);
}
in.close();
System.out.println(response.toString());
```

```
package main

import (
       "bytes"
       "net/http"
)

func main() {

    headers := map[string][]string{
        "Accept": []string{"application/json"},
    }

    data := bytes.NewBuffer([]byte{jsonReq})
    req, err := http.NewRequest("GET", "https://o2r.uni-muenster.de/api/v1/job", data)
    req.Header = headers

    client := &http.Client{}
    resp, err := client.Do(req)
    // ...
}
```

`GET /job`

List Jobs

Lists jobs with filtering and pagination, returning up to 100 results by default.

Results are be sorted by descending date of last change. The content of the response can be limited to certain properties of each result by providing a list of fields, i.e. the parameter `fields`.

Results can be filtered:

- by `compendium_id` i.e. `compendium_id=a4Dnm`,
- by `status` i.e. `status=success` or
- by `user` i.e. `user=0000-0000-0000-0001`

```
curl https://…/api/v1/job?limit=100&start=2&compendium_id=$ID&status=success&fields=status
```

Parameters

Name	In	Type	Required	Description
compendium_id	query	string	false	Comma-separated list of related compendium ids to filter by.
start	query	integer	false	Starting point of the result list. `start - 1` results are skipped. Defaults to `1`.
limit	query	integer	false	Limits the number of results in the response. Defaults to `100`.
user	query	string	false	Public user identifier to filter by.
fields	query	any	false	Specify which additional attributes results list should contain. Can contain following fields: `status`, `user`. Defaults to none.

Detailed descriptions

fields: Specify which additional attributes results list should contain. Can contain following fields: `status`, `user`. Defaults to none.

Enumerated Values

Parameter	Value
fields	status
fields	user
fields	status, user

> Example responses

> `GET /api/v1/job?limit=100&start=2&compendium_id=a4Dnm&status=success`

```
{
  "results": [
    "nkm4L",
    "asdi5",
    "nb2sg",
    "…"
  ]
}
```

> The overall job state can be added to the job list response:

`GET /api/v1/job?limit=100&start=2&compendium_id=a4Dnm&status=success&fields=status`

```
{
  "results": [
```

```
    {
      "id": "nkm4L",
      "status": "failure"
    },
    {
      "id": "asdi5",
      "status": "success"
    },
    {
      "id": "nb2sg",
      "status": "running"
    },
    "…"
  ]
}
```

> If there are no jobs, the returned list is empty.

```
{
  "results": []
}
```

Responses

Status	Meaning	Description	Schema
200	OK	Returns a JSON object with the matching jobs	None

Response Schema

This operation does not require authentication

view_job

> Code samples

```
# You can also use wget
curl -X GET https://o2r.uni-muenster.de/api/v1/job/{job_id} \
  -H 'Accept: application/json'
```

```
GET https://o2r.uni-muenster.de/api/v1/job/{job_id} HTTP/1.1
Host: o2r.uni-muenster.de
Accept: application/json
```

```
const headers = {
  'Accept':'application/json'
};

fetch('https://o2r.uni-muenster.de/api/v1/job/{job_id}',
{
  method: 'GET',

  headers: headers
})
.then(function(res) {
    return res.json();
```

```
}).then(function(body) {
    console.log(body);
});
```

```
require 'rest-client'
require 'json'

headers = {
  'Accept' => 'application/json'
}

result = RestClient.get 'https://o2r.uni-muenster.de/api/v1/job/{job_id}',
  params: {
  }, headers: headers

p JSON.parse(result)
```

```
import requests
headers = {
  'Accept': 'application/json'
}

r = requests.get('https://o2r.uni-muenster.de/api/v1/job/{job_id}', headers = headers)

print(r.json())
```

```
<?php

require 'vendor/autoload.php';

$headers = array(
    'Accept' => 'application/json',
);

$client = new \GuzzleHttp\Client();

// Define array of request body.
$request_body = array();

try {
    $response = $client->request('GET','https://o2r.uni-muenster.de/api/v1/job/{job_id}', array(
        'headers' => $headers,
        'json' => $request_body,
       )
    );
    print_r($response->getBody()->getContents());
 }
 catch (\GuzzleHttp\Exception\BadResponseException $e) {
    // handle exception or api errors.
    print_r($e->getMessage());
 }

 // ...
```

```
URL obj = new URL("https://o2r.uni-muenster.de/api/v1/job/{job_id}");
HttpURLConnection con = (HttpURLConnection) obj.openConnection();
con.setRequestMethod("GET");
int responseCode = con.getResponseCode();
BufferedReader in = new BufferedReader(
    new InputStreamReader(con.getInputStream()));
String inputLine;
StringBuffer response = new StringBuffer();
while ((inputLine = in.readLine()) != null) {
    response.append(inputLine);
```

```
}
in.close();
System.out.println(response.toString());
```

```
package main

import (
       "bytes"
       "net/http"
)

func main() {

    headers := map[string][]string{
        "Accept": []string{"application/json"},
    }

    data := bytes.NewBuffer([]byte{jsonReq})
    req, err := http.NewRequest("GET", "https://o2r.uni-muenster.de/api/v1/job/{job_id}", data)
    req.Header = headers

    client := &http.Client{}
    resp, err := client.Do(req)
    // ...
}
```

`GET /job/{job_id}`

View single job

View details for a single job. The file listing format is described in compendium files.

Step metadata

Additional explanations on steps' status will be transmitted in the `text` property.
`text` is an array, with the latest element holding the newest information.
During long running steps, the `text` element is updated by appending new information when available.

The `start` and `end` timestamps indicate the start and end time of the step. They are formatted as RFC-3339.

Specific steps may carry more information in additional properties.

Step Status

All steps can have one of the following status:

- `queued` : step is not yet started
- `running` : step is currently running
- `success` : step is completed successfully - positive result
- `failure` : step is completed unsuccessfully - negative result
- `skipped` : step does not fit the given input or results of previous steps, e.g. bag validation is not done for non-bag workspaces - neutral result

Parameters

Name	In	Type	Required	Description
steps	query	string	false	The properties `status`, `start` and `end` of *all steps* are always included in the response.

Name	In	Type	Required	Description
job_id	path	string	true	Id of the job

Detailed descriptions

steps: The properties `status`, `start` and `end` of *all steps* are always included in the response.

Supported values for `steps` are `all` or a comma separated list of one or more step names, e.g. `generate_configuration,check`.
The response will contain the default properties for all steps but other properties only for the selected ones.
Any other values for `steps` or not providing the parameter at all will return the default (e.g. `steps=no`).

> Example responses

> ❗ This is an abbreviated example


```
{
  "id": "UMmJ7",
  "compendium_id": "BSgxj",
  "steps": {
    "validate_bag": {
      "status": "skipped",
      "text": [
        "Not a bag"
      ],
      "end": "2017-11-17T13:22:48.105Z",
      "start": "2017-11-17T13:22:48.105Z"
    },
    "generate_configuration": {
      "status": "success",
      "text": [
        "configuration file not found, generating it...",
        "Saved configuration file to job and compendium"
      ],
      "end": "2017-11-17T13:22:48.119Z",
      "start": "2017-11-17T13:22:48.113Z"
    },
    "validate_compendium": {
      "status": "success",
      "text": [
        "all checks passed"
      ],
      "end": "2017-11-17T13:22:48.127Z",
      "start": "2017-11-17T13:22:48.125Z"
    },
    "generate_manifest": {
      "status": "success",
      "text": [
        "INFO [2017-11-17 13:22:56] Going online? TRUE  ... to retrieve system dependencies (sysreq-api)",
        "INFO [2017-11-17 13:22:56] Trying to determine system requirements for the package(s) 'knitr, backpo
        "INFO [2017-11-17 13:22:58] Adding CRAN packages: backports, digest, evaluate, htmltools, knitr, magr
        "INFO [2017-11-17 13:22:58] Created Dockerfile-Object based on /erc/main.Rmd",
        "INFO [2017-11-17 13:22:58] Writing dockerfile to /erc/Dockerfile",
        "generated manifest"
      ],
      "manifest": "Dockerfile",
      "end": "2017-11-17T13:22:58.865Z",
      "start": "2017-11-17T13:22:48.129Z"
    },
    "image_prepare": {
      "status": "success",
      "text": [
        "payload with 756224 total bytes created"
      ],
      "end": "2017-11-17T13:22:58.906Z",
      "start": "2017-11-17T13:22:58.875Z"
```

```
    },
    "image_build": {
      "status": "success",
      "text": [
        "Step 1/6 : FROM rocker/r-ver:3.4.2",
        "---> 3cf05960bf30",
        "---> Running in eb7ccd432592",
        "---> 84db129215f6",
        "Removing intermediate container eb7ccd432592",
        "Successfully built 84db129215f6",
        "Successfully tagged erc:UMmJ7"
      ],
      "end": "2017-11-17T13:22:59.899Z",
      "start": "2017-11-17T13:22:58.912Z"
    },
    "image_execute": {
      "status": "success",
      "text": [
        "[started image execution]",
        "Output created: display.html\r\n> \r\n>",
        "[finished image execution]"
      ],
      "statuscode": 0,
      "start": "2017-11-17T13:22:59.904Z"
    },
    "check": {
      "status": "failure",
      "text": [
        "Check failed"
      ],
      "images": [
        {
          "imageIndex": 0,
          "resizeOperationCode": 0,
          "compareResults": {
            "differences": 204786,
            "dimension": 1290240
          }
        }
      ],
      "display": {
        "diff": "/api/v1/job/UMmJ7/data/check.html"
      },
      "errors": [],
      "checkSuccessful": false,
      "end": "2017-11-17T13:23:04.439Z",
      "start": "2017-11-17T13:23:03.479Z"
    },
    "image_save": {
      "status": "success",
      "text": [
        "[Saving image tarball file]",
        "[Saved image tarball to file (size: 875.14 MB)]"
      ],
      "start": "2018-01-29T17:38:55.111Z",
      "file": "image.tar",
      "end": "2018-01-29T17:39:36.845Z"
    },
    "cleanup": {
      "status": "success",
      "text": [
        "Running regular cleanup",
        "Removed image with tag erc:UMmJ7: [{\"Untagged\":\"erc:UMmJ7\"},{\"Deleted\":\"sha256:84db129215f60f
        "Deleted temporary payload file."
      ],
      "end": "2017-11-17T13:23:05.592Z",
      "start": "2017-11-17T13:23:04.575Z"
    }
  },
  "status": "failure",
  "files": {}
}
```

> Returns an error

```
{
  "error": "no compendium with this id"
}
```

Responses

Status	Meaning	Description	Schema
200	OK	Returns a JSON object with job details.	view_job_response
404	Not Found	No matching compendium found	general_error_model

This operation does not require authentication

Shipment

Ship compendia and metadata

Shipments are used to deliver compendia and their metadata to third party repositories or archives.
This section covers shipment related requests, including repository file management.

Packaging

The packaging of a compendium ensures a recipient can verify the integrity of the transported data.
Currently, the shipment process always creates BagIt bags to package a compendium.

list_reciepients

> Code samples

```
# You can also use wget
curl -X GET https://o2r.uni-muenster.de/api/v1/recipient \
  -H 'Accept: application/json'
```

```
GET https://o2r.uni-muenster.de/api/v1/recipient HTTP/1.1
Host: o2r.uni-muenster.de
Accept: application/json
```

```
const headers = {
  'Accept':'application/json'
};

fetch('https://o2r.uni-muenster.de/api/v1/recipient',
{
  method: 'GET',

  headers: headers
})
.then(function(res) {
```

```
    return res.json();
}).then(function(body) {
    console.log(body);
});
```

```
require 'rest-client'
require 'json'

headers = {
  'Accept' => 'application/json'
}

result = RestClient.get 'https://o2r.uni-muenster.de/api/v1/recipient',
  params: {
  }, headers: headers

p JSON.parse(result)
```

```
import requests
headers = {
  'Accept': 'application/json'
}

r = requests.get('https://o2r.uni-muenster.de/api/v1/recipient', headers = headers)

print(r.json())
```

```
<?php

require 'vendor/autoload.php';

$headers = array(
    'Accept' => 'application/json',
);

$client = new \GuzzleHttp\Client();

// Define array of request body.
$request_body = array();

try {
    $response = $client->request('GET','https://o2r.uni-muenster.de/api/v1/recipient', array(
        'headers' => $headers,
        'json' => $request_body,
       )
    );
    print_r($response->getBody()->getContents());
 }
 catch (\GuzzleHttp\Exception\BadResponseException $e) {
    // handle exception or api errors.
    print_r($e->getMessage());
 }

 // ...
```

```
URL obj = new URL("https://o2r.uni-muenster.de/api/v1/recipient");
HttpURLConnection con = (HttpURLConnection) obj.openConnection();
con.setRequestMethod("GET");
int responseCode = con.getResponseCode();
BufferedReader in = new BufferedReader(
    new InputStreamReader(con.getInputStream()));
String inputLine;
StringBuffer response = new StringBuffer();
while ((inputLine = in.readLine()) != null) {
```

```
    response.append(inputLine);
}
in.close();
System.out.println(response.toString());
```

```
package main

import (
       "bytes"
       "net/http"
)

func main() {

    headers := map[string][]string{
        "Accept": []string{"application/json"},
    }

    data := bytes.NewBuffer([]byte{jsonReq})
    req, err := http.NewRequest("GET", "https://o2r.uni-muenster.de/api/v1/recipient", data)
    req.Header = headers

    client := &http.Client{}
    resp, err := client.Do(req)
    // ...
}
```

`GET /recipient`

Supported recipients

Use the *recipient* endpoint to find out, which repositories are available and configured.
The response is list of tuples with `id` and `label` of each repository.
The `id` is the repository identifier to be used in requests to the `/shipment` endpoint, e.g. to define the recipient, while `label` is a human-readable text string suitable for display in user interfaces.
An implementation may support one or more of the following repositories:

- `b2share` - Eudat b2share
- `b2share_sandbox` - Eudat b2share Sandbox
- `zenodo` - Zenodo Sandbox
- `zenodo_sandbox` - Zenodo Sandbox

The `download` recipient is a surrogate to enable shipping to the user's local storage.

> Example responses

> Retruns A JSON object with all supported recipients.

```
{
  "recipients": [
    {
      "id": "download",
      "label": "Download"
    },
    {
      "id": "b2share_sandbox",
      "label": "Eudat b2share Sandbox"
    },
    {
      "id": "zenodo_sandbox",
      "label": "Zenodo Sandbox"
    }
```

```
  ]
}
```

Responses

Status	Meaning	Description	Schema
200	OK	Retruns A JSON object with all supported recipients.	recipient_response

This operation does not require authentication

list_shipment

> Code samples

```
# You can also use wget
curl -X GET https://o2r.uni-muenster.de/api/v1/shipment \
  -H 'Accept: application/json'
```

```
GET https://o2r.uni-muenster.de/api/v1/shipment HTTP/1.1
Host: o2r.uni-muenster.de
Accept: application/json
```

```
const headers = {
  'Accept':'application/json'
};

fetch('https://o2r.uni-muenster.de/api/v1/shipment',
{
  method: 'GET',

  headers: headers
})
.then(function(res) {
    return res.json();
}).then(function(body) {
    console.log(body);
});
```

```
require 'rest-client'
require 'json'

headers = {
  'Accept' => 'application/json'
}

result = RestClient.get 'https://o2r.uni-muenster.de/api/v1/shipment',
  params: {
  }, headers: headers

p JSON.parse(result)
```

```
import requests
headers = {
  'Accept': 'application/json'
```

```
}

r = requests.get('https://o2r.uni-muenster.de/api/v1/shipment', headers = headers)

print(r.json())
```

```
<?php

require 'vendor/autoload.php';

$headers = array(
    'Accept' => 'application/json',
);

$client = new \GuzzleHttp\Client();

// Define array of request body.
$request_body = array();

try {
    $response = $client->request('GET','https://o2r.uni-muenster.de/api/v1/shipment', array(
        'headers' => $headers,
        'json' => $request_body,
       )
    );
    print_r($response->getBody()->getContents());
 }
 catch (\GuzzleHttp\Exception\BadResponseException $e) {
    // handle exception or api errors.
    print_r($e->getMessage());
 }

 // ...
```

```
URL obj = new URL("https://o2r.uni-muenster.de/api/v1/shipment");
HttpURLConnection con = (HttpURLConnection) obj.openConnection();
con.setRequestMethod("GET");
int responseCode = con.getResponseCode();
BufferedReader in = new BufferedReader(
    new InputStreamReader(con.getInputStream()));
String inputLine;
StringBuffer response = new StringBuffer();
while ((inputLine = in.readLine()) != null) {
    response.append(inputLine);
}
in.close();
System.out.println(response.toString());
```

```
package main

import (
       "bytes"
       "net/http"
)

func main() {

    headers := map[string][]string{
        "Accept": []string{"application/json"},
    }

    data := bytes.NewBuffer([]byte{jsonReq})
    req, err := http.NewRequest("GET", "https://o2r.uni-muenster.de/api/v1/shipment", data)
    req.Header = headers

    client := &http.Client{}
```

```
    resp, err := client.Do(req)
    // ...
}
```

`GET /shipment`

List shipments

This is a basic request to list all shipments identifiers.
`GET /api/v1/shipment`
You can also get only the shipment identifiers belonging to a compendium id (e.g. `4XgD97`).

`GET /api/v1/shipment?compendium_id=4XgD97`

Parameters

Name	In	Type	Required	Description
compendium_id	query	string	false	The identifier of a specific compendium

> Example responses

> Returns a JSON array with all shipment ids

```
[
  "dc351fc6-314f-4947-a235-734ab5971eff",
  "..."
]
```

Responses

Status	Meaning	Description	Schema
200	OK	A list of all shuipment ids.	shipment_response

This operation does not require authentication

view_shipment

> Code samples

```
# You can also use wget
curl -X GET https://o2r.uni-muenster.de/api/v1/shipment/{shipment_id} \
  -H 'Accept: application/json'
```

```
GET https://o2r.uni-muenster.de/api/v1/shipment/{shipment_id} HTTP/1.1
Host: o2r.uni-muenster.de
Accept: application/json
```

```
const headers = {
  'Accept':'application/json'
```

```
};

fetch('https://o2r.uni-muenster.de/api/v1/shipment/{shipment_id}',
{
  method: 'GET',

  headers: headers
})
.then(function(res) {
    return res.json();
}).then(function(body) {
    console.log(body);
});
```

```
require 'rest-client'
require 'json'

headers = {
  'Accept' => 'application/json'
}

result = RestClient.get 'https://o2r.uni-muenster.de/api/v1/shipment/{shipment_id}',
  params: {
  }, headers: headers

p JSON.parse(result)
```

```
import requests
headers = {
  'Accept': 'application/json'
}

r = requests.get('https://o2r.uni-muenster.de/api/v1/shipment/{shipment_id}', headers = headers)

print(r.json())
```

```
<?php

require 'vendor/autoload.php';

$headers = array(
    'Accept' => 'application/json',
);

$client = new \GuzzleHttp\Client();

// Define array of request body.
$request_body = array();

try {
    $response = $client->request('GET','https://o2r.uni-muenster.de/api/v1/shipment/{shipment_id}', array(
        'headers' => $headers,
        'json' => $request_body,
       )
    );
    print_r($response->getBody()->getContents());
 }
 catch (\GuzzleHttp\Exception\BadResponseException $e) {
    // handle exception or api errors.
    print_r($e->getMessage());
 }

 // ...
```

```
URL obj = new URL("https://o2r.uni-muenster.de/api/v1/shipment/{shipment_id}");
HttpURLConnection con = (HttpURLConnection) obj.openConnection();
con.setRequestMethod("GET");
int responseCode = con.getResponseCode();
BufferedReader in = new BufferedReader(
    new InputStreamReader(con.getInputStream()));
String inputLine;
StringBuffer response = new StringBuffer();
while ((inputLine = in.readLine()) != null) {
    response.append(inputLine);
}
in.close();
System.out.println(response.toString());
```

```
package main

import (
       "bytes"
       "net/http"
)

func main() {

    headers := map[string][]string{
        "Accept": []string{"application/json"},
    }

    data := bytes.NewBuffer([]byte{jsonReq})
    req, err := http.NewRequest("GET", "https://o2r.uni-muenster.de/api/v1/shipment/{shipment_id}", data)
    req.Header = headers

    client := &http.Client{}
    resp, err := client.Do(req)
    // ...
}
```

`GET /shipment/{shipment_id}`

Get a single shipment

Example request:

`GET /api/v1/shipment/dc351fc6-314f-4947-a235-734ab5971eff`

Parameters

Name	In	Type	Required	Description
shipment_id	path	string	true	Id of the shipment

> Example responses

> Retruns a JSON object with information on the specified shipment.

⚠ Note

Returned deposition URLs (property `deposition_url`) from Zenodo as well as Eudat b2share (records) will only be functional after publishing.

```
{
  "last_modified": "2016-12-12 10:34:32.001475",
  "recipient": "zenodo",
  "id": "dc351fc6-314f-4947-a235-734ab5971eff",
  "deposition_id": "63179",
  "user": "0000-0002-1825-0097",
  "status": "shipped",
  "compendium_id": "4XgD97",
  "deposition_url": "https://zenodo.org/record/63179"
}
```


Responses

Status	Meaning	Description	Schema
200	OK	A json object with the matching results.	view_shipment_response

This operation does not require authentication

upload_shipment

> Code samples

```
# You can also use wget
curl -X POST https://o2r.uni-muenster.de/api/v1/shipment/{shipment_id} \
  -H 'Content-Type: multipart/form-data' \
  -H 'Accept: application/json'
```

```
POST https://o2r.uni-muenster.de/api/v1/shipment/{shipment_id} HTTP/1.1
Host: o2r.uni-muenster.de
Content-Type: multipart/form-data
Accept: application/json
```

```
const inputBody = '{
  "compendium_id": "string",
  "recipient": "string",
  "update_packaging": false,
  "cookie": "string",
  "shipment_id": "string"
}';
const headers = {
  'Content-Type':'multipart/form-data',
  'Accept':'application/json'
};

fetch('https://o2r.uni-muenster.de/api/v1/shipment/{shipment_id}',
{
  method: 'POST',
  body: inputBody,
  headers: headers
})
.then(function(res) {
    return res.json();
}).then(function(body) {
    console.log(body);
});
```

```
require 'rest-client'
require 'json'

headers = {
  'Content-Type' => 'multipart/form-data',
  'Accept' => 'application/json'
}

result = RestClient.post 'https://o2r.uni-muenster.de/api/v1/shipment/{shipment_id}',
  params: {
  }, headers: headers

p JSON.parse(result)
```

```
import requests
headers = {
  'Content-Type': 'multipart/form-data',
  'Accept': 'application/json'
}

r = requests.post('https://o2r.uni-muenster.de/api/v1/shipment/{shipment_id}', headers = headers)

print(r.json())
```

```
<?php

require 'vendor/autoload.php';

$headers = array(
    'Content-Type' => 'multipart/form-data',
    'Accept' => 'application/json',
);

$client = new \GuzzleHttp\Client();

// Define array of request body.
$request_body = array();

try {
    $response = $client->request('POST','https://o2r.uni-muenster.de/api/v1/shipment/{shipment_id}', array(
        'headers' => $headers,
        'json' => $request_body,
       )
    );
    print_r($response->getBody()->getContents());
 }
 catch (\GuzzleHttp\Exception\BadResponseException $e) {
    // handle exception or api errors.
    print_r($e->getMessage());
 }

 // ...
```

```
URL obj = new URL("https://o2r.uni-muenster.de/api/v1/shipment/{shipment_id}");
HttpURLConnection con = (HttpURLConnection) obj.openConnection();
con.setRequestMethod("POST");
int responseCode = con.getResponseCode();
BufferedReader in = new BufferedReader(
    new InputStreamReader(con.getInputStream()));
String inputLine;
StringBuffer response = new StringBuffer();
while ((inputLine = in.readLine()) != null) {
    response.append(inputLine);
}
```

```
in.close();
System.out.println(response.toString());
```

```
package main

import (
       "bytes"
       "net/http"
)

func main() {

    headers := map[string][]string{
        "Content-Type": []string{"multipart/form-data"},
        "Accept": []string{"application/json"},
    }

    data := bytes.NewBuffer([]byte{jsonReq})
    req, err := http.NewRequest("POST", "https://o2r.uni-muenster.de/api/v1/shipment/{shipment_id}", data)
    req.Header = headers

    client := &http.Client{}
    resp, err := client.Do(req)
    // ...
}
```

`POST /shipment/{shipment_id}`

Create a new shipment

You can start a initial creation of a shipment, leading to transmission to a repository and creation of a deposition, using a `POST` request.

`POST /api/v1/shipment`

⚠ Required user level

The user sending the request to create a shipment must have the required [user level](user/levels.md).

> Body parameter

```
compendium_id: string
recipient: string
update_packaging: false
cookie: string
shipment_id: string
```

Parameters

Name	In	Type	Required	Description
body	body	new_shipment	true	The shipment to be uploaded.
shipment_id	path	string	true	Id of the shipment

Detailed descriptions

body: The shipment to be uploaded.

> Example responses

> The response contains the shipment document, see Get a single shipment. Some of the fields are not available (have value `null`) until after publishing, e.g. `deposition_url`.

```
{
  "id": "9ff3d75e-23dc-423e-a6c6-6987ac5ffc3e",
  "recipient": "zenodo",
  "status": "shipped",
  "deposition_id": "79102"
}
```

> If the recipient is the download surrogate, the response will be `202` and a zip stream with the Content type `application/zip`. The download zip stream is also available under the url of the shipment plus /dl, once it has been created, e.g.:

`http://localhost:8087/api/v1/shipment/22e7b17c-0047-4cb9-9041-bb87f30de388/dl`

Responses

Status	Meaning	Description	Schema
201	Created	The response contains the shipment document, see Get a single shipment. Some of the fields are not available (have value `null`) until after publishing, e.g. `deposition_url`.	upload_shipment_response
202	Accepted	If the recipient is the download surrogate, the response will be `202` and a zip stream with the Content type `application/zip`. The download zip stream is also available under the url of the shipment plus /dl, once it has been created, e.g.:	

`http://localhost:8087/api/v1/shipment/22e7b17c-0047-4cb9-9041-bb87f30de388/dl` |None|

Response Schema

To perform this operation, you must be authenticated by means of one of the following methods: cookie_authenification

shipment_status

> Code samples

```
# You can also use wget
curl -X GET https://o2r.uni-muenster.de/api/v1/shipment/{shipment_id}/status \
  -H 'Accept: application/json'
```

```
GET https://o2r.uni-muenster.de/api/v1/shipment/{shipment_id}/status HTTP/1.1
Host: o2r.uni-muenster.de
Accept: application/json
```

```
const headers = {
  'Accept':'application/json'
```

```
};

fetch('https://o2r.uni-muenster.de/api/v1/shipment/{shipment_id}/status',
{
  method: 'GET',

  headers: headers
})
.then(function(res) {
    return res.json();
}).then(function(body) {
    console.log(body);
});
```

```
require 'rest-client'
require 'json'

headers = {
  'Accept' => 'application/json'
}

result = RestClient.get 'https://o2r.uni-muenster.de/api/v1/shipment/{shipment_id}/status',
  params: {
  }, headers: headers

p JSON.parse(result)
```

```
import requests
headers = {
  'Accept': 'application/json'
}

r = requests.get('https://o2r.uni-muenster.de/api/v1/shipment/{shipment_id}/status', headers = headers)

print(r.json())
```

```
<?php

require 'vendor/autoload.php';

$headers = array(
    'Accept' => 'application/json',
);

$client = new \GuzzleHttp\Client();

// Define array of request body.
$request_body = array();

try {
    $response = $client->request('GET','https://o2r.uni-muenster.de/api/v1/shipment/{shipment_id}/status', ar
        'headers' => $headers,
        'json' => $request_body,
       )
    );
    print_r($response->getBody()->getContents());
 }
 catch (\GuzzleHttp\Exception\BadResponseException $e) {
    // handle exception or api errors.
    print_r($e->getMessage());
 }

 // ...
```

```
URL obj = new URL("https://o2r.uni-muenster.de/api/v1/shipment/{shipment_id}/status");
HttpURLConnection con = (HttpURLConnection) obj.openConnection();
con.setRequestMethod("GET");
int responseCode = con.getResponseCode();
BufferedReader in = new BufferedReader(
    new InputStreamReader(con.getInputStream()));
String inputLine;
StringBuffer response = new StringBuffer();
while ((inputLine = in.readLine()) != null) {
    response.append(inputLine);
}
in.close();
System.out.println(response.toString());
```

```
package main

import (
       "bytes"
       "net/http"
)

func main() {

    headers := map[string][]string{
        "Accept": []string{"application/json"},
    }

    data := bytes.NewBuffer([]byte{jsonReq})
    req, err := http.NewRequest("GET", "https://o2r.uni-muenster.de/api/v1/shipment/{shipment_id}/status", da
    req.Header = headers

    client := &http.Client{}
    resp, err := client.Do(req)
    // ...
}
```

`GET /shipment/{shipment_id}/status`

Shipment status

A shipment can have three possible status:

- `shipped` - a deposition has been created at a repository and completed the necessary metadata for publication.
- `published` - the contents of the shipment are published on the repository, in which case the publishment can not be undone.
- `error` - an error occurred during shipment or publishing.

To get only a shipment's current status you may use the sub-resource `/status`.

Parameters

Name	In	Type	Required	Description
shipment_id	path	string	true	Id of the shipment

> Example responses

> Retruns A JSON object with the shipment id and status.

```
{
  "id": "9ff3d75e-23dc-423e-a6c6-6987ac5ffc3e",
  "status": "shipped"
}
```

Responses

Status	Meaning	Description	Schema
200	OK	Retruns A JSON object with the shipment id and status.	shipment_status_response

This operation does not require authentication

publish_deposition

> Code samples

```
# You can also use wget
curl -X PUT https://o2r.uni-muenster.de/api/v1/shipment/{shipment_id}/publishment \
  -H 'Accept: application/json'
```

```
PUT https://o2r.uni-muenster.de/api/v1/shipment/{shipment_id}/publishment HTTP/1.1
Host: o2r.uni-muenster.de
Accept: application/json
```

```
const headers = {
  'Accept':'application/json'
};

fetch('https://o2r.uni-muenster.de/api/v1/shipment/{shipment_id}/publishment',
{
  method: 'PUT',

  headers: headers
})
.then(function(res) {
    return res.json();
}).then(function(body) {
    console.log(body);
});
```

```
require 'rest-client'
require 'json'

headers = {
  'Accept' => 'application/json'
}

result = RestClient.put 'https://o2r.uni-muenster.de/api/v1/shipment/{shipment_id}/publishment',
  params: {
  }, headers: headers

p JSON.parse(result)
```

```python
import requests
headers = {
  'Accept': 'application/json'
}

r = requests.put('https://o2r.uni-muenster.de/api/v1/shipment/{shipment_id}/publishment', headers = headers)

print(r.json())
```

```php
<?php

require 'vendor/autoload.php';

$headers = array(
    'Accept' => 'application/json',
);

$client = new \GuzzleHttp\Client();

// Define array of request body.
$request_body = array();

try {
    $response = $client->request('PUT','https://o2r.uni-muenster.de/api/v1/shipment/{shipment_id}/publishment
        'headers' => $headers,
        'json' => $request_body,
       )
    );
    print_r($response->getBody()->getContents());
 }
 catch (\GuzzleHttp\Exception\BadResponseException $e) {
    // handle exception or api errors.
    print_r($e->getMessage());
 }

 // ...
```

```java
URL obj = new URL("https://o2r.uni-muenster.de/api/v1/shipment/{shipment_id}/publishment");
HttpURLConnection con = (HttpURLConnection) obj.openConnection();
con.setRequestMethod("PUT");
int responseCode = con.getResponseCode();
BufferedReader in = new BufferedReader(
    new InputStreamReader(con.getInputStream()));
String inputLine;
StringBuffer response = new StringBuffer();
while ((inputLine = in.readLine()) != null) {
    response.append(inputLine);
}
in.close();
System.out.println(response.toString());
```

```go
package main

import (
       "bytes"
       "net/http"
)

func main() {

    headers := map[string][]string{
        "Accept": []string{"application/json"},
    }

    data := bytes.NewBuffer([]byte{jsonReq})
```

```
    req, err := http.NewRequest("PUT", "https://o2r.uni-muenster.de/api/v1/shipment/{shipment_id}/publishment
    req.Header = headers

    client := &http.Client{}
    resp, err := client.Do(req)
    // ...
}
```

`PUT /shipment/{shipment_id}/publishment`

Publish in a deposition

The publishment is supposed to have completed the status shipped where metadata requirements for publication have been checked.

⚠ Note

Once published, a deposition can no longer be deleted on the supported repositories.

`PUT api/v1/shipment/<shipment_id>/publishment`

Note that a publishment is not possible if the recipient is the download surrogate which immediately results in a zip stream as a response.

Parameters

Name	In	Type	Required	Description
shipment_id	path	string	true	Id of the shipment

> Example responses

> Returns a JSON object with the id and status of the shipment

```
{
  "id": "9ff3d75e-23dc-423e-a6c6-6987ac5ffc3e",
  "status": "published"
}
```

Responses

Status	Meaning	Description	Schema
200	OK	Returns a JSON object with the id and status of the shipment	shipment_status_response

This operation does not require authentication

Bindings

Bindings

A binding is an optional component of an Executable Research Compendium.
It can be used to make static figures interactive, for example, to show how different parameter settings affect the result shown in the figure.
A bindings stores information on the code lines needed to generate the corresponding figure, the parameter that should be made interactive, the data subset required for the figure, and the user interface (UI) widget (e.g. a slider or radio buttons).
The resulting JSON object including the binding is stored in the metadata tag `interaction`.
For this reason, the compendium must be at least a `candidate` where the metadata extraction is completed.

upload_binding

> Code samples

```
# You can also use wget
curl -X POST https://o2r.uni-muenster.de/api/v1/bindings/binding \
  -H 'Content-Type: application/json' \
  -H 'Accept: application/json'
```

```
POST https://o2r.uni-muenster.de/api/v1/bindings/binding HTTP/1.1
Host: o2r.uni-muenster.de
Content-Type: application/json
Accept: application/json
```

```
const inputBody = '{
  "id": "rDdFN",
  "computationalResult": {
    "type": "figure",
    "result": "Figure 3"
  },
  "sourcecode": {
    "file": "main.Rmd",
    "codelines": [
      {
        "first_line": 101,
        "last_line": 503
      }
    ],
    "parameter": [
      {
        "text": "duration <- 24",
        "name": "duration",
        "val": 24,
        "uiWidget": {
          "minValue": 1,
          "type": "slider",
          "maxValue": 24,
          "stepSize": 1,
          "caption": "The duration parameter specifies the duration of the flood event in hours."
        }
      }
    ]
  }
}';
const headers = {
  'Content-Type':'application/json',
  'Accept':'application/json'
};

fetch('https://o2r.uni-muenster.de/api/v1/bindings/binding',
{
  method: 'POST',
  body: inputBody,
  headers: headers
})
```

```
.then(function(res) {
    return res.json();
}).then(function(body) {
    console.log(body);
});
```

```
require 'rest-client'
require 'json'

headers = {
  'Content-Type' => 'application/json',
  'Accept' => 'application/json'
}

result = RestClient.post 'https://o2r.uni-muenster.de/api/v1/bindings/binding',
  params: {
  }, headers: headers

p JSON.parse(result)
```

```
import requests
headers = {
  'Content-Type': 'application/json',
  'Accept': 'application/json'
}

r = requests.post('https://o2r.uni-muenster.de/api/v1/bindings/binding', headers = headers)

print(r.json())
```

```
<?php

require 'vendor/autoload.php';

$headers = array(
    'Content-Type' => 'application/json',
    'Accept' => 'application/json',
);

$client = new \GuzzleHttp\Client();

// Define array of request body.
$request_body = array();

try {
    $response = $client->request('POST','https://o2r.uni-muenster.de/api/v1/bindings/binding', array(
        'headers' => $headers,
        'json' => $request_body,
       )
    );
    print_r($response->getBody()->getContents());
 }
 catch (\GuzzleHttp\Exception\BadResponseException $e) {
    // handle exception or api errors.
    print_r($e->getMessage());
 }

 // ...
```

```
URL obj = new URL("https://o2r.uni-muenster.de/api/v1/bindings/binding");
HttpURLConnection con = (HttpURLConnection) obj.openConnection();
con.setRequestMethod("POST");
int responseCode = con.getResponseCode();
BufferedReader in = new BufferedReader(
```

```
    new InputStreamReader(con.getInputStream()));
String inputLine;
StringBuffer response = new StringBuffer();
while ((inputLine = in.readLine()) != null) {
    response.append(inputLine);
}
in.close();
System.out.println(response.toString());
```

```
package main

import (
       "bytes"
       "net/http"
)

func main() {

    headers := map[string][]string{
        "Content-Type": []string{"application/json"},
        "Accept": []string{"application/json"},
    }

    data := bytes.NewBuffer([]byte{jsonReq})
    req, err := http.NewRequest("POST", "https://o2r.uni-muenster.de/api/v1/bindings/binding", data)
    req.Header = headers

    client := &http.Client{}
    resp, err := client.Do(req)
    // ...
}
```

`POST /bindings/binding`

Create new binding

> Body parameter

```
{
  "id": "rDdFN",
  "computationalResult": {
    "type": "figure",
    "result": "Figure 3"
  },
  "sourcecode": {
    "file": "main.Rmd",
    "codelines": [
      {
        "first_line": 101,
        "last_line": 503
      }
    ],
    "parameter": [
      {
        "text": "duration <- 24",
        "name": "duration",
        "val": 24,
        "uiWidget": {
          "minValue": 1,
          "type": "slider",
          "maxValue": 24,
          "stepSize": 1,
          "caption": "The duration parameter specifies the duration of the flood event in hours."
        }
      }
    ]
```

```
  }
}
```

Parameters

Name	In	Type	Required	Description
body	body	new_binding	true	The Bindning to be uploaded.

Detailed descriptions

body: The Bindning to be uploaded.

> Example responses

> Binding Upload response

```
{
  "callback": "ok",
  "data": "binding"
}
```

Responses

Status	Meaning	Description	Schema
200	OK	Binding Upload response	upload_binding_response

This operation does not require authentication

proxy_binding

> Code samples

```
# You can also use wget
curl -X GET https://o2r.uni-muenster.de/api/v1/compendium/{compendium_id}/binding/{binding_id}
```

```
GET https://o2r.uni-muenster.de/api/v1/compendium/{compendium_id}/binding/{binding_id} HTTP/1.1
Host: o2r.uni-muenster.de
```

```
fetch('https://o2r.uni-muenster.de/api/v1/compendium/{compendium_id}/binding/{binding_id}',
{
  method: 'GET'

})
.then(function(res) {
    return res.json();
}).then(function(body) {
    console.log(body);
});
```

```ruby
require 'rest-client'
require 'json'

result = RestClient.get 'https://o2r.uni-muenster.de/api/v1/compendium/{compendium_id}/binding/{binding_id}',
  params: {
  }

p JSON.parse(result)
```

```python
import requests

r = requests.get('https://o2r.uni-muenster.de/api/v1/compendium/{compendium_id}/binding/{binding_id}')

print(r.json())
```

```php
<?php

require 'vendor/autoload.php';

$client = new \GuzzleHttp\Client();

// Define array of request body.
$request_body = array();

try {
    $response = $client->request('GET','https://o2r.uni-muenster.de/api/v1/compendium/{compendium_id}/binding
        'headers' => $headers,
        'json' => $request_body,
       )
    );
    print_r($response->getBody()->getContents());
 }
 catch (\GuzzleHttp\Exception\BadResponseException $e) {
    // handle exception or api errors.
    print_r($e->getMessage());
 }

 // ...
```

```java
URL obj = new URL("https://o2r.uni-muenster.de/api/v1/compendium/{compendium_id}/binding/{binding_id}");
HttpURLConnection con = (HttpURLConnection) obj.openConnection();
con.setRequestMethod("GET");
int responseCode = con.getResponseCode();
BufferedReader in = new BufferedReader(
    new InputStreamReader(con.getInputStream()));
String inputLine;
StringBuffer response = new StringBuffer();
while ((inputLine = in.readLine()) != null) {
    response.append(inputLine);
}
in.close();
System.out.println(response.toString());
```

```go
package main

import (
       "bytes"
       "net/http"
)

func main() {
```

```
    data := bytes.NewBuffer([]byte{jsonReq})
    req, err := http.NewRequest("GET", "https://o2r.uni-muenster.de/api/v1/compendium/{compendium_id}/binding
    req.Header = headers

    client := &http.Client{}
    resp, err := client.Do(req)
    // ...
}
```

`GET /compendium/{compendium_id}/binding/{binding_id}`

Proxy for binding

Parameters

Name	In	Type	Required	Description
compendium_id	path	string	true	Id of the compendium
binding_id	path	string	true	Id of the binding

Responses

Status	Meaning	Description	Schema
200	OK	Successfull Proxy	None

This operation does not require authentication

run_binding

> Code samples

```
# You can also use wget
curl -X POST https://o2r.uni-muenster.de/api/v1/compendium/{compendium_id}/binding/{binding_id}
```

```
POST https://o2r.uni-muenster.de/api/v1/compendium/{compendium_id}/binding/{binding_id} HTTP/1.1
Host: o2r.uni-muenster.de
```

```
fetch('https://o2r.uni-muenster.de/api/v1/compendium/{compendium_id}/binding/{binding_id}',
{
  method: 'POST'

})
.then(function(res) {
    return res.json();
}).then(function(body) {
    console.log(body);
});
```

```
require 'rest-client'
require 'json'
```

```
result = RestClient.post 'https://o2r.uni-muenster.de/api/v1/compendium/{compendium_id}/binding/{binding_id}'
  params: {
  }

p JSON.parse(result)
```

```
import requests

r = requests.post('https://o2r.uni-muenster.de/api/v1/compendium/{compendium_id}/binding/{binding_id}')

print(r.json())
```

```
<?php

require 'vendor/autoload.php';

$client = new \GuzzleHttp\Client();

// Define array of request body.
$request_body = array();

try {
    $response = $client->request('POST','https://o2r.uni-muenster.de/api/v1/compendium/{compendium_id}/bindin
        'headers' => $headers,
        'json' => $request_body,
       )
    );
    print_r($response->getBody()->getContents());
 }
 catch (\GuzzleHttp\Exception\BadResponseException $e) {
    // handle exception or api errors.
    print_r($e->getMessage());
 }

 // ...
```

```
URL obj = new URL("https://o2r.uni-muenster.de/api/v1/compendium/{compendium_id}/binding/{binding_id}");
HttpURLConnection con = (HttpURLConnection) obj.openConnection();
con.setRequestMethod("POST");
int responseCode = con.getResponseCode();
BufferedReader in = new BufferedReader(
    new InputStreamReader(con.getInputStream()));
String inputLine;
StringBuffer response = new StringBuffer();
while ((inputLine = in.readLine()) != null) {
    response.append(inputLine);
}
in.close();
System.out.println(response.toString());
```

```
package main

import (
       "bytes"
       "net/http"
)

func main() {

    data := bytes.NewBuffer([]byte{jsonReq})
    req, err := http.NewRequest("POST", "https://o2r.uni-muenster.de/api/v1/compendium/{compendium_id}/bindin
    req.Header = headers

    client := &http.Client{}
```

```
    resp, err := client.Do(req)
    // ...
}
```

`POST /compendium/{compendium_id}/binding/{binding_id}`

Run binding

Parameters

Name	In	Type	Required	Description
compendium_id	path	string	true	Id of the compendium
binding_id	path	string	true	Id of the binding

Responses

Status	Meaning	Description	Schema
200	OK	Successfull run of binding	None

This operation does not require authentication

search_binding

> Code samples

```
# You can also use wget
curl -X POST https://o2r.uni-muenster.de/api/v1/bindings/searchBinding \
  -H 'Content-Type: application/json' \
  -H 'Accept: application/json'
```

```
POST https://o2r.uni-muenster.de/api/v1/bindings/searchBinding HTTP/1.1
Host: o2r.uni-muenster.de
Content-Type: application/json
Accept: application/json
```

```
const inputBody = '{
  "term": "processData(input)",
  "metadata": "{o2r.metadata}"
}';
const headers = {
  'Content-Type':'application/json',
  'Accept':'application/json'
};

fetch('https://o2r.uni-muenster.de/api/v1/bindings/searchBinding',
{
  method: 'POST',
  body: inputBody,
  headers: headers
})
.then(function(res) {
    return res.json();
}).then(function(body) {
```

```
    console.log(body);
});
```

```
require 'rest-client'
require 'json'

headers = {
  'Content-Type' => 'application/json',
  'Accept' => 'application/json'
}

result = RestClient.post 'https://o2r.uni-muenster.de/api/v1/bindings/searchBinding',
  params: {
  }, headers: headers

p JSON.parse(result)
```

```
import requests
headers = {
  'Content-Type': 'application/json',
  'Accept': 'application/json'
}

r = requests.post('https://o2r.uni-muenster.de/api/v1/bindings/searchBinding', headers = headers)

print(r.json())
```

```
<?php

require 'vendor/autoload.php';

$headers = array(
    'Content-Type' => 'application/json',
    'Accept' => 'application/json',
);

$client = new \GuzzleHttp\Client();

// Define array of request body.
$request_body = array();

try {
    $response = $client->request('POST','https://o2r.uni-muenster.de/api/v1/bindings/searchBinding', array(
        'headers' => $headers,
        'json' => $request_body,
       )
    );
    print_r($response->getBody()->getContents());
 }
 catch (\GuzzleHttp\Exception\BadResponseException $e) {
    // handle exception or api errors.
    print_r($e->getMessage());
 }

 // ...
```

```
URL obj = new URL("https://o2r.uni-muenster.de/api/v1/bindings/searchBinding");
HttpURLConnection con = (HttpURLConnection) obj.openConnection();
con.setRequestMethod("POST");
int responseCode = con.getResponseCode();
BufferedReader in = new BufferedReader(
    new InputStreamReader(con.getInputStream()));
String inputLine;
StringBuffer response = new StringBuffer();
```

```
while ((inputLine = in.readLine()) != null) {
    response.append(inputLine);
}
in.close();
System.out.println(response.toString());
```

```
package main

import (
       "bytes"
       "net/http"
)

func main() {

    headers := map[string][]string{
        "Content-Type": []string{"application/json"},
        "Accept": []string{"application/json"},
    }

    data := bytes.NewBuffer([]byte{jsonReq})
    req, err := http.NewRequest("POST", "https://o2r.uni-muenster.de/api/v1/bindings/searchBinding", data)
    req.Header = headers

    client := &http.Client{}
    resp, err := client.Do(req)
    // ...
}
```

`POST /bindings/searchBinding`

Search for a binding

> Body parameter

```
{
  "term": "processData(input)",
  "metadata": "{o2r.metadata}"
}
```

Parameters

Name	In	Type	Required	Description
body	body	search_binding	true	Request body for finding bindings that include a certain code snippet.

Detailed descriptions

body: Request body for finding bindings that include a certain code snippet.

> Example responses

> Search Binding response

```
{
  "callback": "ok",
  "data": [
    "result"
```

```
    ]
  }
```

Responses

Status	Meaning	Description	Schema
200	OK	Search Binding response	search_binding_response

This operation does not require authentication

Users

User levels

Users are authenticated via OAuth and the actions on the website are limited by the `level` associated with an account.
On registration, each account is assigned a level `0`.
Only admin users and the user herself can read the level of a user.

The following is a list of actions and the corresponding required *minimum* user level.

- `0` *Users* (everybody)
  - Create new jobs
  - View compendia, jobs, user details
- `100` *Known users*
  - Create new compendium
  - Create shipments
  - Create substitutions
  - Delete own candidates
- `500` *Editors*
  - Edit user levels up to own level
  - Edit compendium metadata
  - Delete candidates
  - Manage public links for candidates
- `1000` *Admins*
  - Edit user levels up to own level
  - Delete compendia and candidates
  - View status pages of microservices

User authentication

User authentication is done via authenticated sessions, which are referenced with a cookie called `connect.sid`. For every endpoint that needs user authentication, a cookie with an authenticated session is required.

Client authentication

To execute restricted operations of the API, such as [compendium upload]() or [job execution](), a client must provide an authentication token via a cookie.

A client must first login on the website to access a browser cookie issued by `o2r.uni-muenster.de` with the name `connect.sid`.
Provide the content of the cookie when making requests to the API as shown in the request example below.

Access authentication information for direct API access

To run commands which require authentication from the command line, a user must login on the website first.
Then open you browser cookies and find a cookie issued by `o2r.uni-muenster.de` with the name `connect.sid`.
Use the the contents of the cookie for your requests, for example as shown below when using curl.

```
curl [...] --cookie "connect.sid=<code string here>" \
     https://…/api/v1/endpoint
```

Authentication within microservices

Attention: The authentication process *requires* a secured connection, i.e. `HTTPS`.

Authentication provider

Session authentication is done using the OAuth 2.0 protocol.
Currently ORCID is the only available authentication provider, therefore users need to be registered with ORCID. Because of its nature, the authentication workflow is not a RESTful service.
Users must follow the redirection to the login endpoint with their web browser and grant access to the o2r reproducibility service for their ORCID account.
They are then sent back to our authentication service, which verifies the authentication request and enriches the user session with the verified ORCID for this user.

list_user

> Code samples

```
# You can also use wget
curl -X GET https://o2r.uni-muenster.de/api/v1/user \
  -H 'Accept: application/json'
```

```
GET https://o2r.uni-muenster.de/api/v1/user HTTP/1.1
Host: o2r.uni-muenster.de
Accept: application/json
```

```
const headers = {
  'Accept':'application/json'
};

fetch('https://o2r.uni-muenster.de/api/v1/user',
{
  method: 'GET',

  headers: headers
})
.then(function(res) {
    return res.json();
}).then(function(body) {
    console.log(body);
});
```

```
require 'rest-client'
require 'json'
```

```
headers = {
  'Accept' => 'application/json'
}

result = RestClient.get 'https://o2r.uni-muenster.de/api/v1/user',
  params: {
  }, headers: headers

p JSON.parse(result)
```

```
import requests
headers = {
  'Accept': 'application/json'
}

r = requests.get('https://o2r.uni-muenster.de/api/v1/user', headers = headers)

print(r.json())
```

```
<?php

require 'vendor/autoload.php';

$headers = array(
    'Accept' => 'application/json',
);

$client = new \GuzzleHttp\Client();

// Define array of request body.
$request_body = array();

try {
    $response = $client->request('GET','https://o2r.uni-muenster.de/api/v1/user', array(
        'headers' => $headers,
        'json' => $request_body,
       )
    );
    print_r($response->getBody()->getContents());
 }
 catch (\GuzzleHttp\Exception\BadResponseException $e) {
    // handle exception or api errors.
    print_r($e->getMessage());
 }

 // ...
```

```
URL obj = new URL("https://o2r.uni-muenster.de/api/v1/user");
HttpURLConnection con = (HttpURLConnection) obj.openConnection();
con.setRequestMethod("GET");
int responseCode = con.getResponseCode();
BufferedReader in = new BufferedReader(
    new InputStreamReader(con.getInputStream()));
String inputLine;
StringBuffer response = new StringBuffer();
while ((inputLine = in.readLine()) != null) {
    response.append(inputLine);
}
in.close();
System.out.println(response.toString());
```

```
package main
```

```
import (
       "bytes"
       "net/http"
)

func main() {

    headers := map[string][]string{
        "Accept": []string{"application/json"},
    }

    data := bytes.NewBuffer([]byte{jsonReq})
    req, err := http.NewRequest("GET", "https://o2r.uni-muenster.de/api/v1/user", data)
    req.Header = headers

    client := &http.Client{}
    resp, err := client.Do(req)
    // ...
}
```

`GET /user`

List users

Return a list of user ids. Pagination (including defaults) as described for compendia is available for users.

`curl https://…/api/v1/user`

`GET /api/v1/user`

Parameters

Name	In	Type	Required	Description
start	query	integer	false	`start` is the index of the first list item in the response, defaults to `1`. It must be numeric and larger than `0`.
limit	query	integer	false	`limit` is the number of results in the response, defaults to `100`. It numeric and larger than `0`.

Detailed descriptions

start: `start` is the index of the first list item in the response, defaults to `1`. It must be numeric and larger than `0`.

limit: `limit` is the number of results in the response, defaults to `100`. It numeric and larger than `0`.

> Example responses

> Returns a JSON object with user ids.

```
{
  "results": [
    "0000-0002-1825-0097",
    "0000-0002-1825-0097"
  ]
}
```

> If there are no users, the returned list is empty.

```
{
  "results": []
}
```

> Bad Request

```
{
  "error": "limit must be larger than 0"
}
```

Responses

Status	Meaning	Description	Schema
200	OK	User listing response	list_response
400	Bad Request	Bad Request	general_error_model

This operation does not require authentication

view_user

> Code samples

```
# You can also use wget
curl -X GET https://o2r.uni-muenster.de/api/v1/user/{user_id} \
  -H 'Accept: application/json'
```

```
GET https://o2r.uni-muenster.de/api/v1/user/{user_id} HTTP/1.1
Host: o2r.uni-muenster.de
Accept: application/json
```

```
const headers = {
  'Accept':'application/json'
};

fetch('https://o2r.uni-muenster.de/api/v1/user/{user_id}',
{
  method: 'GET',

  headers: headers
})
.then(function(res) {
    return res.json();
}).then(function(body) {
    console.log(body);
});
```

```
require 'rest-client'
require 'json'

headers = {
  'Accept' => 'application/json'
```

```
}

result = RestClient.get 'https://o2r.uni-muenster.de/api/v1/user/{user_id}',
  params: {
  }, headers: headers

p JSON.parse(result)
```

```
import requests
headers = {
  'Accept': 'application/json'
}

r = requests.get('https://o2r.uni-muenster.de/api/v1/user/{user_id}', headers = headers)

print(r.json())
```

```
<?php

require 'vendor/autoload.php';

$headers = array(
    'Accept' => 'application/json',
);

$client = new \GuzzleHttp\Client();

// Define array of request body.
$request_body = array();

try {
    $response = $client->request('GET','https://o2r.uni-muenster.de/api/v1/user/{user_id}', array(
        'headers' => $headers,
        'json' => $request_body,
       )
    );
    print_r($response->getBody()->getContents());
 }
 catch (\GuzzleHttp\Exception\BadResponseException $e) {
    // handle exception or api errors.
    print_r($e->getMessage());
 }

 // ...
```

```
URL obj = new URL("https://o2r.uni-muenster.de/api/v1/user/{user_id}");
HttpURLConnection con = (HttpURLConnection) obj.openConnection();
con.setRequestMethod("GET");
int responseCode = con.getResponseCode();
BufferedReader in = new BufferedReader(
    new InputStreamReader(con.getInputStream()));
String inputLine;
StringBuffer response = new StringBuffer();
while ((inputLine = in.readLine()) != null) {
    response.append(inputLine);
}
in.close();
System.out.println(response.toString());
```

```
package main

import (
       "bytes"
       "net/http"
```

```
)

func main() {

    headers := map[string][]string{
        "Accept": []string{"application/json"},
    }

    data := bytes.NewBuffer([]byte{jsonReq})
    req, err := http.NewRequest("GET", "https://o2r.uni-muenster.de/api/v1/user/{user_id}", data)
    req.Header = headers

    client := &http.Client{}
    resp, err := client.Do(req)
    // ...
}
```

`GET /user/{user_id}`

View a single user

Show the details of a user.

`curl https://…/api/v1/user/$ID`

`GET /api/v1/user/:id`

Parameters

Name	In	Type	Required	Description
user_id	path	string	true	Id of the user

> Example responses

> View user response

```
{
  "id": "0000-0002-1825-0097",
  "name": "o2r"
}
```

> The content of the response depends on the state and level of the user that requests the resource. The above response only contains the id and the publicly visible name. The following response contains more details and requires a certain user level of the authenticated user making the request:

`curl --cookie "connect.sid=<session cookie here>" https://…/api/v1/user/0000-0002-1825-0097`

```
{
  "id": "0000-0002-1825-0097",
  "name": "o2r",
  "level": 0,
  "lastseen": "2016-08-15T12:32:23.972Z"
}
```

> Not found

```
{
  "error": "no user with this id"
}
```

Responses

Status	Meaning	Description	Schema
200	OK	View user response	user_view_response
404	Not Found	Not found	general_error_model

This operation does not require authentication

edit_user

> Code samples

```
# You can also use wget
curl -X PATCH https://o2r.uni-muenster.de/api/v1/user/{user_id} \
  -H 'Accept: application/json'
```

```
PATCH https://o2r.uni-muenster.de/api/v1/user/{user_id} HTTP/1.1
Host: o2r.uni-muenster.de
Accept: application/json
```

```
const headers = {
  'Accept':'application/json'
};

fetch('https://o2r.uni-muenster.de/api/v1/user/{user_id}',
{
  method: 'PATCH',

  headers: headers
})
.then(function(res) {
    return res.json();
}).then(function(body) {
    console.log(body);
});
```

```
require 'rest-client'
require 'json'

headers = {
  'Accept' => 'application/json'
}

result = RestClient.patch 'https://o2r.uni-muenster.de/api/v1/user/{user_id}',
  params: {
  }, headers: headers

p JSON.parse(result)
```

```
import requests
headers = {
  'Accept': 'application/json'
}

r = requests.patch('https://o2r.uni-muenster.de/api/v1/user/{user_id}', headers = headers)

print(r.json())
```

```
<?php

require 'vendor/autoload.php';

$headers = array(
    'Accept' => 'application/json',
);

$client = new \GuzzleHttp\Client();

// Define array of request body.
$request_body = array();

try {
    $response = $client->request('PATCH','https://o2r.uni-muenster.de/api/v1/user/{user_id}', array(
        'headers' => $headers,
        'json' => $request_body,
       )
    );
    print_r($response->getBody()->getContents());
 }
 catch (\GuzzleHttp\Exception\BadResponseException $e) {
    // handle exception or api errors.
    print_r($e->getMessage());
 }

 // ...
```

```
URL obj = new URL("https://o2r.uni-muenster.de/api/v1/user/{user_id}");
HttpURLConnection con = (HttpURLConnection) obj.openConnection();
con.setRequestMethod("PATCH");
int responseCode = con.getResponseCode();
BufferedReader in = new BufferedReader(
    new InputStreamReader(con.getInputStream()));
String inputLine;
StringBuffer response = new StringBuffer();
while ((inputLine = in.readLine()) != null) {
    response.append(inputLine);
}
in.close();
System.out.println(response.toString());
```

```
package main

import (
       "bytes"
       "net/http"
)

func main() {

    headers := map[string][]string{
        "Accept": []string{"application/json"},
    }

    data := bytes.NewBuffer([]byte{jsonReq})
```

```
    req, err := http.NewRequest("PATCH", "https://o2r.uni-muenster.de/api/v1/user/{user_id}", data)
    req.Header = headers

    client := &http.Client{}
    resp, err := client.Do(req)
    // ...
}
```

`PATCH /user/{user_id}`

Edit user

You can update information of an existing user using the `HTTP` operation `PATCH`.
Change user level request

The user level can be changed with an `HTTP` `PATCH` request. The new level is passed to the API via a query parameter, i.e. ..?level=. The value must be an int (integer). The response is the full user document with the updated value.

⚠ Required user level

The user sending the request to change the level must have the required user level

Parameters

Name	In	Type	Required	Description
level	query	integer	false	none
user_id	path	string	true	Id of the user

> Example responses

> `curl --request PATCH --cookie "connect.sid=<session cookie here>"` \
> `https://…/api/v1/user/0000-0002-1825-0097?level=42`

```
{
  "id": "0000-0002-1825-0097",
  "name": "o2r",
  "level": 42,
  "lastseen": "2016-08-15T12:32:23.972Z"
}
```

> Bad request

```
{
  "error": "parameter 'level' could not be parsed as an integer"
}
```

> Unauthorized

```
{
  "error": "user is not authenticated"
}
```

> User

```
{
  "error": "user level does not allow edit"
}
```

Not found

```
{
  "error": "no user with this id"
}
```

Responses

Status	Meaning	Description	Schema
200	OK	Edit user response	user_view_response
400	Bad Request	Bad request	general_error_model
401	Unauthorized	Unauthorized	general_error_model
403	Forbidden	User	general_error_model
404	Not Found	Not found	general_error_model

This operation does not require authentication

login_user

Code samples

```
# You can also use wget
curl -X GET https://o2r.uni-muenster.de/api/v1/auth/login
```

```
GET https://o2r.uni-muenster.de/api/v1/auth/login HTTP/1.1
Host: o2r.uni-muenster.de
```

```
fetch('https://o2r.uni-muenster.de/api/v1/auth/login',
{
  method: 'GET'

})
.then(function(res) {
    return res.json();
}).then(function(body) {
    console.log(body);
});
```

```
require 'rest-client'
require 'json'

result = RestClient.get 'https://o2r.uni-muenster.de/api/v1/auth/login',
  params: {
```

```
  }

p JSON.parse(result)
```

```
import requests

r = requests.get('https://o2r.uni-muenster.de/api/v1/auth/login')

print(r.json())
```

```
<?php

require 'vendor/autoload.php';

$client = new \GuzzleHttp\Client();

// Define array of request body.
$request_body = array();

try {
    $response = $client->request('GET','https://o2r.uni-muenster.de/api/v1/auth/login', array(
        'headers' => $headers,
        'json' => $request_body,
       )
    );
    print_r($response->getBody()->getContents());
 }
 catch (\GuzzleHttp\Exception\BadResponseException $e) {
    // handle exception or api errors.
    print_r($e->getMessage());
 }

 // ...
```

```
URL obj = new URL("https://o2r.uni-muenster.de/api/v1/auth/login");
HttpURLConnection con = (HttpURLConnection) obj.openConnection();
con.setRequestMethod("GET");
int responseCode = con.getResponseCode();
BufferedReader in = new BufferedReader(
    new InputStreamReader(con.getInputStream()));
String inputLine;
StringBuffer response = new StringBuffer();
while ((inputLine = in.readLine()) != null) {
    response.append(inputLine);
}
in.close();
System.out.println(response.toString());
```

```
package main

import (
       "bytes"
       "net/http"
)

func main() {

    data := bytes.NewBuffer([]byte{jsonReq})
    req, err := http.NewRequest("GET", "https://o2r.uni-muenster.de/api/v1/auth/login", data)
    req.Header = headers

    client := &http.Client{}
    resp, err := client.Do(req)
```

```
    // ...
}
```

`GET /auth/login`

Start OAuth login

Navigate the web browser (e.g. via a HTML `<a>` link) to `/api/v1/auth/login`, which then redirects the user and request access to your ORCID profile. After granting access, ORCID redirects the user back to the `/api/v1/auth/login` endpoint with a unique `code` param that is used to verify the request.

If the verification was successful, the endpoint returns a session cookie named `connect.sid`, which is tied to a authenticated session. The server answers with a `302 redirect`, which redirects the user back to `/`, the start page of the o2r website.

If the login is unsuccessful, the user is not redirected back to the site and no further redirects are configured.

Responses

Status	Meaning	Description	Schema
302	Found	Access request for OCRID profile then it will redirect to the Main Page	None

This operation does not require authentication

whoami_user

> Code samples

```
# You can also use wget
curl -X GET https://o2r.uni-muenster.de/api/v1/auth/whoami \
  -H 'Accept: application/json'
```

```
GET https://o2r.uni-muenster.de/api/v1/auth/whoami HTTP/1.1
Host: o2r.uni-muenster.de
Accept: application/json
```

```
const headers = {
  'Accept':'application/json'
};

fetch('https://o2r.uni-muenster.de/api/v1/auth/whoami',
{
  method: 'GET',

  headers: headers
})
.then(function(res) {
    return res.json();
}).then(function(body) {
    console.log(body);
});
```

```
require 'rest-client'
require 'json'
```

```
headers = {
  'Accept' => 'application/json'
}

result = RestClient.get 'https://o2r.uni-muenster.de/api/v1/auth/whoami',
  params: {
  }, headers: headers

p JSON.parse(result)
```

```
import requests
headers = {
  'Accept': 'application/json'
}

r = requests.get('https://o2r.uni-muenster.de/api/v1/auth/whoami', headers = headers)

print(r.json())
```

```
<?php

require 'vendor/autoload.php';

$headers = array(
    'Accept' => 'application/json',
);

$client = new \GuzzleHttp\Client();

// Define array of request body.
$request_body = array();

try {
    $response = $client->request('GET','https://o2r.uni-muenster.de/api/v1/auth/whoami', array(
        'headers' => $headers,
        'json' => $request_body,
       )
    );
    print_r($response->getBody()->getContents());
 }
 catch (\GuzzleHttp\Exception\BadResponseException $e) {
    // handle exception or api errors.
    print_r($e->getMessage());
 }

 // ...
```

```
URL obj = new URL("https://o2r.uni-muenster.de/api/v1/auth/whoami");
HttpURLConnection con = (HttpURLConnection) obj.openConnection();
con.setRequestMethod("GET");
int responseCode = con.getResponseCode();
BufferedReader in = new BufferedReader(
    new InputStreamReader(con.getInputStream()));
String inputLine;
StringBuffer response = new StringBuffer();
while ((inputLine = in.readLine()) != null) {
    response.append(inputLine);
}
in.close();
System.out.println(response.toString());
```

```
package main
```

```
import (
       "bytes"
       "net/http"
)

func main() {

    headers := map[string][]string{
        "Accept": []string{"application/json"},
    }

    data := bytes.NewBuffer([]byte{jsonReq})
    req, err := http.NewRequest("GET", "https://o2r.uni-muenster.de/api/v1/auth/whoami", data)
    req.Header = headers

    client := &http.Client{}
    resp, err := client.Do(req)
    // ...
}
```

`GET /auth/whoami`

Request authentication status

As the cookie is present in both authenticated and unauthenticated sessions, clients (e.g. web browser user interfaces) must know if their session is authenticated, and if so, as which ORCID user. For this, send a `GET` request to the `/api/v1/auth/whoami` endpoint, including your session cookie.

```
curl https://…/api/v1/auth/whoami --cookie "connect.sid=…
```

`GET /api/v1/auth/whoami`

> Example responses

> Returns ocrid id and user name as a JSON object

```
{
  "orcid": "0000-0002-1825-0097",
  "name": "o2r"
}
```

> Unauthorized

```
{
  "error": "user is not authenticated"
}
```

Responses

Status	Meaning	Description	Schema
200	OK	Returns ocrid id and user name as a JSON object	user_whoami_response
401	Unauthorized	Unauthorized	general_error_model

To perform this operation, you must be authenticated by means of one of the following methods: cookie_authenification

API Info

Get technical info from the api

get_api_info

> Code samples

```
# You can also use wget
curl -X GET https://o2r.uni-muenster.de/api/v1/api \
  -H 'Accept: application/json'
```

```
GET https://o2r.uni-muenster.de/api/v1/api HTTP/1.1
Host: o2r.uni-muenster.de
Accept: application/json
```

```
const headers = {
  'Accept':'application/json'
};

fetch('https://o2r.uni-muenster.de/api/v1/api',
{
  method: 'GET',

  headers: headers
})
.then(function(res) {
    return res.json();
}).then(function(body) {
    console.log(body);
});
```

```
require 'rest-client'
require 'json'

headers = {
  'Accept' => 'application/json'
}

result = RestClient.get 'https://o2r.uni-muenster.de/api/v1/api',
  params: {
  }, headers: headers

p JSON.parse(result)
```

```
import requests
headers = {
  'Accept': 'application/json'
}

r = requests.get('https://o2r.uni-muenster.de/api/v1/api', headers = headers)

print(r.json())
```

```php
<?php

require 'vendor/autoload.php';

$headers = array(
    'Accept' => 'application/json',
);

$client = new \GuzzleHttp\Client();

// Define array of request body.
$request_body = array();

try {
    $response = $client->request('GET','https://o2r.uni-muenster.de/api/v1/api', array(
        'headers' => $headers,
        'json' => $request_body,
       )
    );
    print_r($response->getBody()->getContents());
 }
 catch (\GuzzleHttp\Exception\BadResponseException $e) {
    // handle exception or api errors.
    print_r($e->getMessage());
 }

 // ...
```

```java
URL obj = new URL("https://o2r.uni-muenster.de/api/v1/api");
HttpURLConnection con = (HttpURLConnection) obj.openConnection();
con.setRequestMethod("GET");
int responseCode = con.getResponseCode();
BufferedReader in = new BufferedReader(
    new InputStreamReader(con.getInputStream()));
String inputLine;
StringBuffer response = new StringBuffer();
while ((inputLine = in.readLine()) != null) {
    response.append(inputLine);
}
in.close();
System.out.println(response.toString());
```

```go
package main

import (
       "bytes"
       "net/http"
)

func main() {

    headers := map[string][]string{
        "Accept": []string{"application/json"},
    }

    data := bytes.NewBuffer([]byte{jsonReq})
    req, err := http.NewRequest("GET", "https://o2r.uni-muenster.de/api/v1/api", data)
    req.Header = headers

    client := &http.Client{}
    resp, err := client.Do(req)
    // ...
}
```

`GET /api`

Get api info

> Example responses

> Returns JSON object with some api info

```
{
  "about": "https://o2r.info",
  "versions": {
    "current": "/api/v1",
    "v1": "/api/v1"
  },
  "quote": "There's always a bigger fish."
}
```

Responses

Status	Meaning	Description	Schema
200	OK	Returns JSON object with some api info	api_info_response

This operation does not require authentication

get_api_list

> Code samples

```
# You can also use wget
curl -X GET https://o2r.uni-muenster.de/api/v1/api/v1 \
  -H 'Accept: application/json'
```

```
GET https://o2r.uni-muenster.de/api/v1/api/v1 HTTP/1.1
Host: o2r.uni-muenster.de
Accept: application/json
```

```
const headers = {
  'Accept':'application/json'
};

fetch('https://o2r.uni-muenster.de/api/v1/api/v1',
{
  method: 'GET',

  headers: headers
})
.then(function(res) {
    return res.json();
}).then(function(body) {
    console.log(body);
});
```

```
require 'rest-client'
require 'json'

headers = {
```

```
  'Accept' => 'application/json'
}

result = RestClient.get 'https://o2r.uni-muenster.de/api/v1/api/v1',
  params: {
  }, headers: headers

p JSON.parse(result)
```

```
import requests
headers = {
  'Accept': 'application/json'
}

r = requests.get('https://o2r.uni-muenster.de/api/v1/api/v1', headers = headers)

print(r.json())
```

```
<?php

require 'vendor/autoload.php';

$headers = array(
    'Accept' => 'application/json',
);

$client = new \GuzzleHttp\Client();

// Define array of request body.
$request_body = array();

try {
    $response = $client->request('GET','https://o2r.uni-muenster.de/api/v1/api/v1', array(
        'headers' => $headers,
        'json' => $request_body,
       )
    );
    print_r($response->getBody()->getContents());
 }
 catch (\GuzzleHttp\Exception\BadResponseException $e) {
    // handle exception or api errors.
    print_r($e->getMessage());
 }

 // ...
```

```
URL obj = new URL("https://o2r.uni-muenster.de/api/v1/api/v1");
HttpURLConnection con = (HttpURLConnection) obj.openConnection();
con.setRequestMethod("GET");
int responseCode = con.getResponseCode();
BufferedReader in = new BufferedReader(
    new InputStreamReader(con.getInputStream()));
String inputLine;
StringBuffer response = new StringBuffer();
while ((inputLine = in.readLine()) != null) {
    response.append(inputLine);
}
in.close();
System.out.println(response.toString());
```

```
package main

import (
       "bytes"
```

```
        "net/http"
)

func main() {

    headers := map[string][]string{
        "Accept": []string{"application/json"},
    }

    data := bytes.NewBuffer([]byte{jsonReq})
    req, err := http.NewRequest("GET", "https://o2r.uni-muenster.de/api/v1/api/v1", data)
    req.Header = headers

    client := &http.Client{}
    resp, err := client.Do(req)
    // ...
}
```

`GET /api/v1`

List api endpoints

> Example responses

> Returns JSON object with the currently available endpoints.


```
{
  "auth": "/api/v1/auth",
  "compendia": "/api/v1/compendium",
  "jobs": "/api/v1/job",
  "users": "/api/v1/user",
  "search": "/api/v1/search",
  "shipments": "/api/v1/shipment",
  "recipients": "/api/v1/recipient",
  "substitutions": "/api/v1/substitution",
  "links": "/api/v1/link"
}
```


Responses

Status	Meaning	Description	Schema
200	OK	Returns JSON object with the currently available endpoints.	api_list_response

This operation does not require authentication

Schemas

new_compendium


```
{
  "content_type": "compendium",
  "compendium": "string",
  "path": "string",
```

```
  "doi": "string",
  "zenodo_record_id": "string",
  "filename": "string"
}
```

Properties

Name	Type	Required	Restrictions	Description
content_type	string	true	none	The Type of archive that is to be uploaded. Can be `compendium` or `workspace`. - `compendium` - compendium, which is expected to be complete and valid, for *examination* of a compendium - `workspace` - formless workspace, for *creation* of a compendium **⚠ Warning** If a complete ERC is submitted as a workspace, it may result in an error, or the contained metadata and other files may be overwritten by the creation process.
compendium	string(binary)	false	none	The archive file as a compressed .zip
path	string	false	none	Path to a subdirectory or a zip file in the public share (optional) - default is `/` - the leading `/` is optional, the loader supports both ways - when a directory has multiple zip files, the path can be used to specify which file is used, e.g. `path=/metatainer.zip`
doi	string	false	none	A DOI resolving to the zenodo record
zenodo_record_id	string	false	none	The ID of the zenodo record
filename	string	false	none	Filename of your compendium. For now, only zip-files are supported. - if no `filename` is provided the first zip file is selected - multiple files are currently not supported

Enumerated Values

Property	Value
content_type	compendium
content_type	workspace

new_job

```
{
  "compendium_id": "string"
}
```

Properties

Name	Type	Required	Restrictions	Description
compendium_id	string	true	none	The identifier of the compendium to base this job on.

new_shipment

```
{
  "compendium_id": "string",
  "recipient": "string",
  "update_packaging": false,
  "cookie": "string",
  "shipment_id": "string"
}
```

Properties

Name	Type	Required	Restrictions	Description
compendium_id	string	true	none	The id of the compendium
recipient	string	true	none	The id of the compendium
update_packaging	boolean	false	none	The shipment creation only succeeds if a valid package is already present under the provided compendium identifier, or if no packaging is present at all and a new package can be created. In case a partial or invalid package is given, this parameter can control the shipment creation process: If it is set to `true`, the shipment package is updated during the shipment creation in order to make it valid, if set to `false` the shipment creation results in an error.
cookie	string	false	none	An authentication cookie must be set in the request header, but it may also be provided via a `cookie` form parameter as a fallback

Name	Type	Required	Restrictions	Description
shipment_id	string	false	none	A user-defined identifier for the shipment (see id in response)

new_substitution

```
{
  "base": "string",
  "overlay": "string",
  "substitutionFiles": [
    {
      "base": "string",
      "overlay": "string"
    }
  ],
  "metadataHandling": "keepBase"
}
```

Properties

Name	Type	Required	Restrictions	Description
base	string	false	none	id of the base compendium
overlay	string	false	none	id of the overlay compendium
substitutionFiles	[object]	false	none	Array of file substitutions specified by `base` and `overlay`
» base	string	false	none	name of the file from base compendium
» overlay	string	false	none	name of the overlay compendium that is exchanged for the original file
metadataHandling	string	false	none	property to specify, if the metadata of the base ERC will be adopted (`keepBase` = **keep metadata** of base ERC) or there will be a new extraction of metadata, that will be merged into the metadata of the base ERC (`extractAndMerge` = **extract and merge metadata** for new ERC) or that will not be merged (`extract` = **extract metadata** of new ERC)

Enumerated Values

Property	Value
metadataHandling	keepBase
metadataHandling	extractAndMerge

Property	Value
metadataHandling	extract

new_search

```
{
  "query": {
    "bool": {
      "must": {},
      "filter": {}
    }
  },
  "from": 0,
  "size": 0
}
```

Properties

Name	Type	Required	Restrictions	Description
query	object	false	none	none
» bool	object	false	none	none
»» must	object	false	none	none
»» filter	object	false	none	none
from	number	false	none	none
size	integer	false	none	none

new_binding

```
{
  "id": "string",
  "computationalResult": {
    "type": "string",
    "result": "string"
  },
  "sourcecode": {
    "file": "string",
    "codelines": [
      {
        "first_line": 0,
        "last_line": 0
      }
    ],
    "parameter": [
```

```
      {
        "text": "string",
        "name": "string",
        "val": 0,
        "uiWidget": {
          "type": "string",
          "minValue": 0,
          "maxValue": 0,
          "stepSize": 0,
          "caption": "string"
        }
      }
    ]
  }
}
```

Properties

Name	Type	Required	Restrictions	Description
id	string	false	none	ID of the compendium for which the bindings should be created
computationalResult	object	false	none	specification of the computational result that should be made interactive
» type	string	false	none	type of the computational result, e.g. a figure or a table
» result	string	false	none	the actual result as it is referred to in the text
sourcecode	object	false	none	all code-related information needed to create a binding for result specified above
» file	string	false	none	main file of the research compendium containing the R code
» codelines	[object]	false	none	array of code chunks including the code needed to generate the result
»» first_line	integer	false	none	start of the code chunk
»» last_line	integer	false	none	end of the code chunk
» parameter	[object]	false	none	array of parameters that should be made interactive
»» text	string	false	none	parameter as it is initialized in the code
»» name	string	false	none	the name of the parameter without the value
»» val	integer	false	none	the value of the parameter without the name
»» uiWidget	object	false	none	specification of the UI widget, one per parameter, here examplified with a slider
»»» type	string	false	none	widget type, e.g. a slider or radio buttons
»»» minValue	integer	false	none	minimum value of the slider
»»» maxValue	integer	false	none	maximum value of the slider
»»» stepSize	integer	false	none	step size when moving the slider

Name	Type	Required	Restrictions	Description
»»» caption	string	false	none	caption for the result

extract_binding

```
{
  "id": "string",
  "file": "string",
  "plot": "string"
}
```

Properties

Name	Type	Required	Restrictions	Description
id	string	false	none	ID of the compendium for which the bindings should be created
file	string	false	none	main file of the research compendium containing the R code
plot	string	false	none	function that outputs the result used as starting point for the backtracking algorithm

search_binding

```
{
  "term": "string",
  "metadata": "string"
}
```

Properties

Name	Type	Required	Restrictions	Description
term	string	false	none	search for bindings including this code snippet
metadata	string	false	none	whole metadata object of the corresponding compendium

upload_response

```
{
  "id": "string"
}
```


Properties

Name	Type	Required	Restrictions	Description
id	string	false	none	none

list_response


```
{
  "results": [
    "string"
  ]
}
```


Properties

Name	Type	Required	Restrictions	Description
results	[string]	false	none	none

view_response


```
{
  "id": "string",
  "metadata": {},
  "created": "string",
  "user": "string",
  "bag": true,
  "compendium": true,
  "substituted": true,
  "files": {}
}
```


Properties

Name	Type	Required	Restrictions	Description
id	user_id	false	none	none
metadata	object	false	none	none

Name	Type	Required	Restrictions	Description
created	string(date-string)	false	none	none
user	string	false	none	none
bag	boolean	false	none	none
compendium	boolean	false	none	none
substituted	boolean	false	none	none
files	object	false	none	none

metadata_response


```
{
  "id": "string",
  "metadata": {
    "o2r": {}
  }
}
```


Properties

Name	Type	Required	Restrictions	Description
id	string	false	none	none
metadata	object	false	none	none
» o2r	object	false	none	none

job_response


```
{
  "job_id": "string"
}
```


Properties

Name	Type	Required	Restrictions	Description
job_id	string	false	none	none

view_job_response

```
{
  "id": "string",
  "compendium_id": "string",
  "steps:": {},
  "status": "success",
  "files": {}
}
```

Properties

Name	Type	Required	Restrictions	Description
id	string	false	none	none
compendium_id	string	false	none	none
steps:	object	false	none	none
status	string	false	none	none
files	object	false	none	none

Enumerated Values

Property	Value
status	success
status	failure
status	running

recipient_response

```
{
  "recipients": [
    {
      "id": "string",
      "label": "string"
    }
  ]
}
```

Properties

Name	Type	Required	Restrictions	Description
recipients	[object]	false	none	none
» id	string	false	none	none
» label	string	false	none	none

shipment_response

```
[
  "string"
]
```

Properties

None

view_shipment_response

```
{
  "id": "string",
  "compendium_id": "string",
  "deposition_id": "string",
  "deposition_url": "string",
  "update_packaging": "string",
  "recipient": "string",
  "last_modified": "2019-08-24T14:15:22Z",
  "user": "string",
  "status": "shipped",
  "md": {}
}
```

Properties

Name	Type	Required	Restrictions	Description
id	string	false	none	none
compendium_id	string	false	none	none
deposition_id	string	false	none	none
deposition_url	string	false	none	none
update_packaging	string	false	none	none

Name	Type	Required	Restrictions	Description
recipient	string	false	none	none
last_modified	string(date-time)	false	none	none
user	string	false	none	none
status	string	false	none	none
md	object	false	none	none

Enumerated Values

Property	Value
status	shipped
status	published
status	error

upload_shipment_response

```
{
  "id": "string",
  "deposition_id": "string",
  "recipient": "string",
  "status": "shipped"
}
```

Properties

Name	Type	Required	Restrictions	Description
id	string	false	none	none
deposition_id	string	false	none	none
recipient	string	false	none	none
status	string	false	none	none

Enumerated Values

Property	Value
status	shipped
status	published

Property	Value
status	error

shipment_status_response

```
{
  "id": "string",
  "status": "shipped"
}
```

Properties

Name	Type	Required	Restrictions	Description
id	string	false	none	none
status	string	false	none	none

Enumerated Values

Property	Value
status	shipped
status	published
status	error

shipment_files_response

```
{
  "files": [
    {}
  ]
}
```

Properties

Name	Type	Required	Restrictions	Description
files	[object]	false	none	none

link_response

```
{
  "id": "string",
  "compendium_id": "string",
  "user": "string"
}
```

Properties

Name	Type	Required	Restrictions	Description
id	string	false	none	none
compendium_id	string	false	none	none
user	string(user_id)	false	none	none

link_list_response

```
{
  "results": [
    {
      "id": "string",
      "compendium_id": "string",
      "user": "string"
    }
  ]
}
```

Properties

Name	Type	Required	Restrictions	Description
results	[object]	false	none	none
» id	string	false	none	none
» compendium_id	string	false	none	none
» user	string(user_id)	false	none	none

upload_binding_response

```
{
  "callback": "string",
  "data": null
}
```

Properties

Name	Type	Required	Restrictions	Description
callback	string	false	none	none
data	any	false	none	binding

search_binding_response

```
{
  "callback": "string",
  "data": [
    "string"
  ]
}
```

Properties

Name	Type	Required	Restrictions	Description
callback	string	false	none	none
data	[string]	false	none	array of results for which bindings exist that include the corresponding code snippet

user_view_response

```
{
  "id": "string",
  "name": "string",
  "level": 0,
  "lastseen": "string"
}
```

Properties

Name	Type	Required	Restrictions	Description
id	string(user_id)	true	none	none
name	string	true	none	none
level	integer	false	none	none
lastseen	string(datetime)	false	none	none

user_whoami_response

```
{
  "ocrid": "string",
  "name": "string"
}
```

Properties

Name	Type	Required	Restrictions	Description
ocrid	string	false	none	none
name	string	false	none	none

api_info_response

```
{
  "about": "string",
  "versions": {},
  "quote": "string"
}
```

Properties

Name	Type	Required	Restrictions	Description
about	string	false	none	none
versions	object	false	none	none
quote	string	false	none	none

api_list_response

```
{}
```

Properties

None

simple_search_response

```
{
  "hits": {
    "total": 0,
    "max_score": 0,
    "hits": [
      {}
    ]
  }
}
```

Properties

Name	Type	Required	Restrictions	Description
hits	object	false	none	none
» total	integer	false	none	none
» max_score	number	false	none	none
» hits	[object]	false	none	none

general_error_model

```
{
  "error": "string"
}
```

Properties

Name	Type	Required	Restrictions	Description

Name	Type	Required	Restrictions	Description
error	string	false	none	none

user_id

```
"string"
```

Properties

Name	Type	Required	Restrictions	Description
anonymous	string	false	none	none