B2SHARE API hands-on

Publish your data in B2SHARE using Python

Hans van Piggelen | SURFsara

Today's B2SHARE hands-on

- Introduction: (30 min)
  - The B2SHARE REST API
  - B2SHARE concepts, variables, metadata schemas
  - In-depth: making requests, authentication and payloads
  - Publication workflow

- Hands-on: (1 hour)
  - Simple examples
  - Hands-on exercises using **Jupyter Notebook**

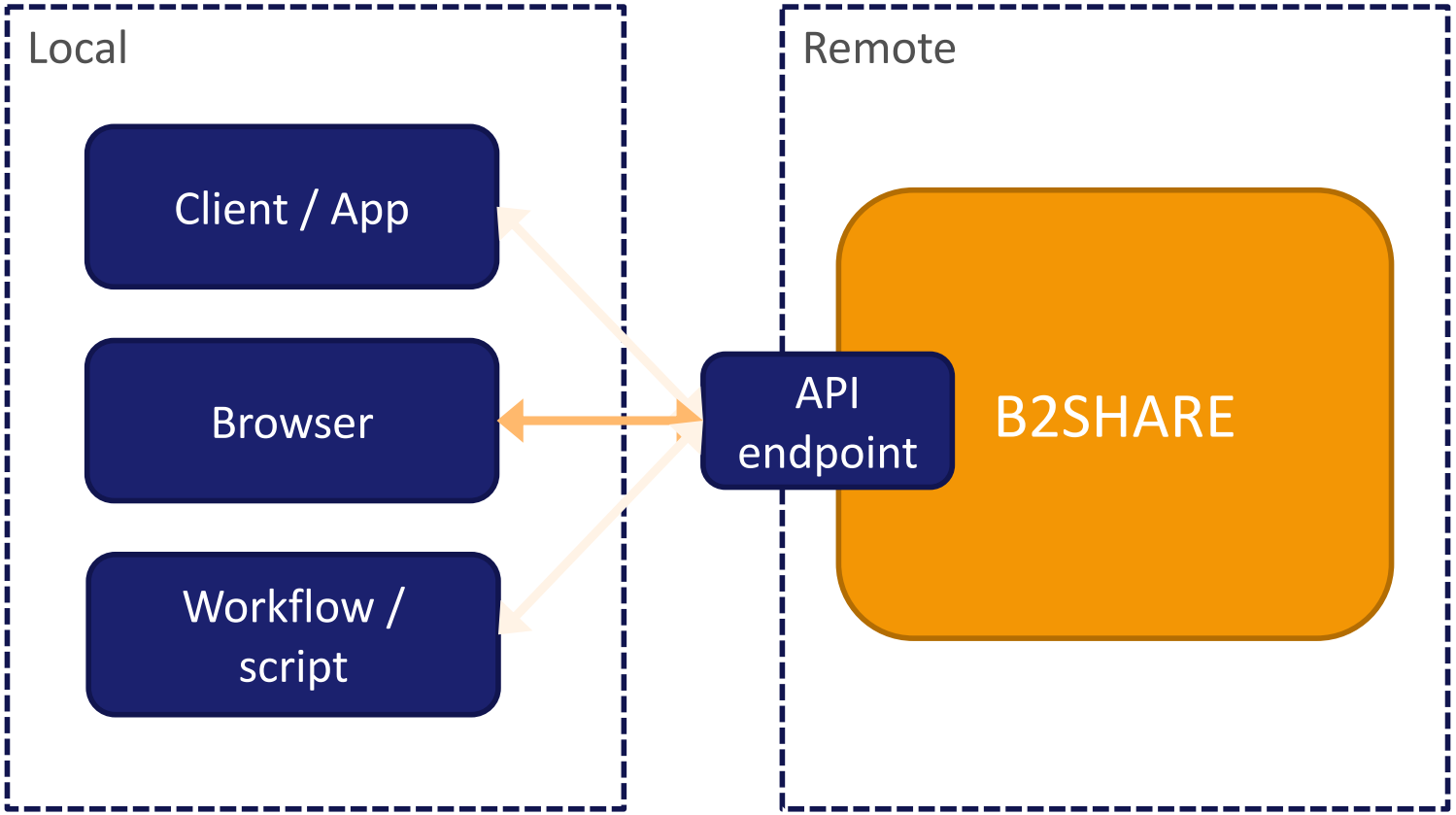
B2SHARE

- B2SHARE is about data:
  - Storage
  - Sharing
  - Preservation
  - Accessibility
  - Publication
  - Curation
- But also:
  - Citation
  - Metadata
  - Checksums
  - Linking
  - Management

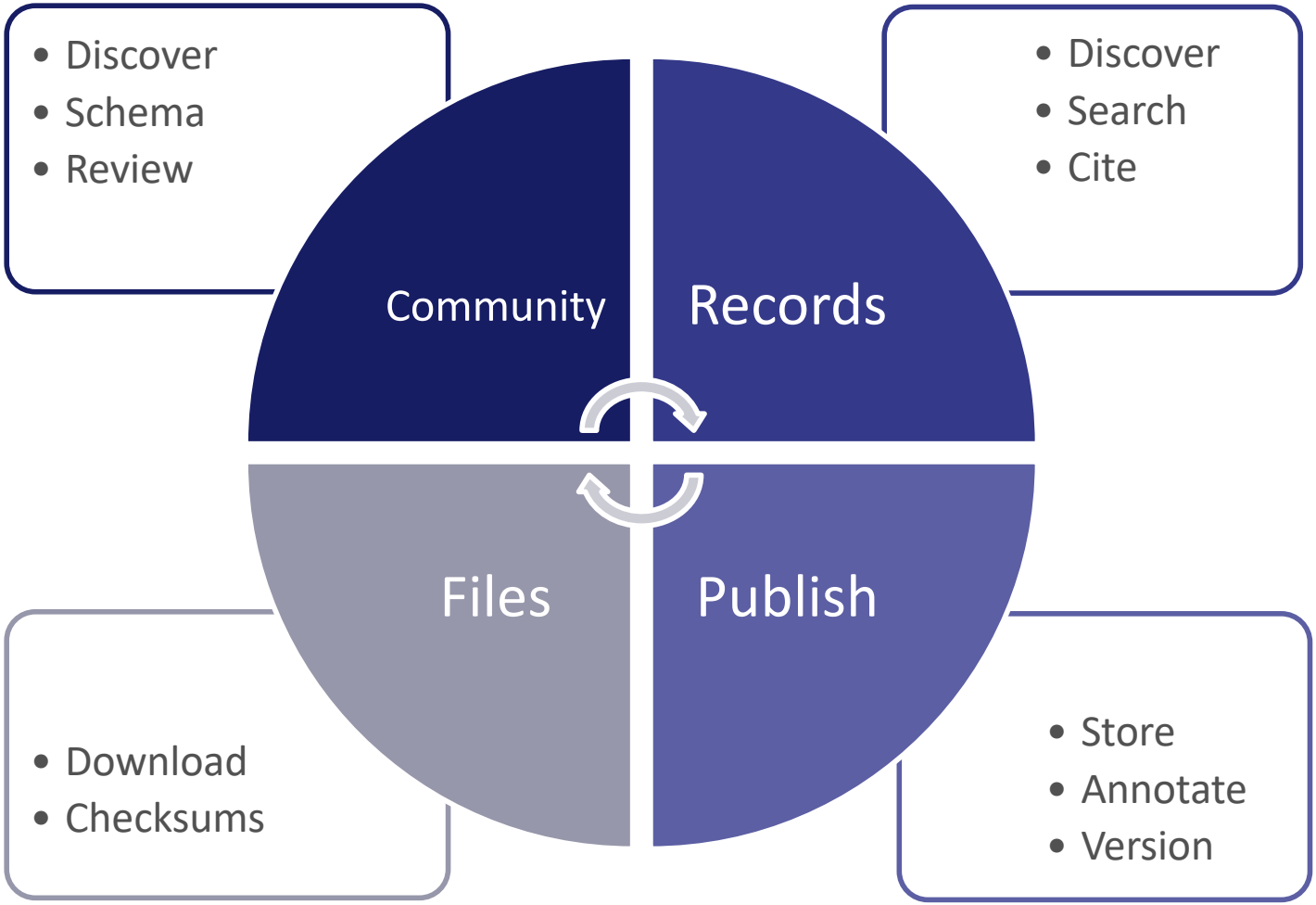
Most importantly : FAIR

What is the B2SHARE API?

- Set of instructions for **interaction** with B2SHARE
- Implemented as **HTTP RESTful API**



What can I do with the B2SHARE API?



Why use the B2SHARE API?

- Make your life easier now and on the long term!

Automate	Quality	Management
• Creation • File upload • Annotation • Finalisation • Retrieval	• Exact metadata • Ingest • Review	• Integration • Workflows • Overview

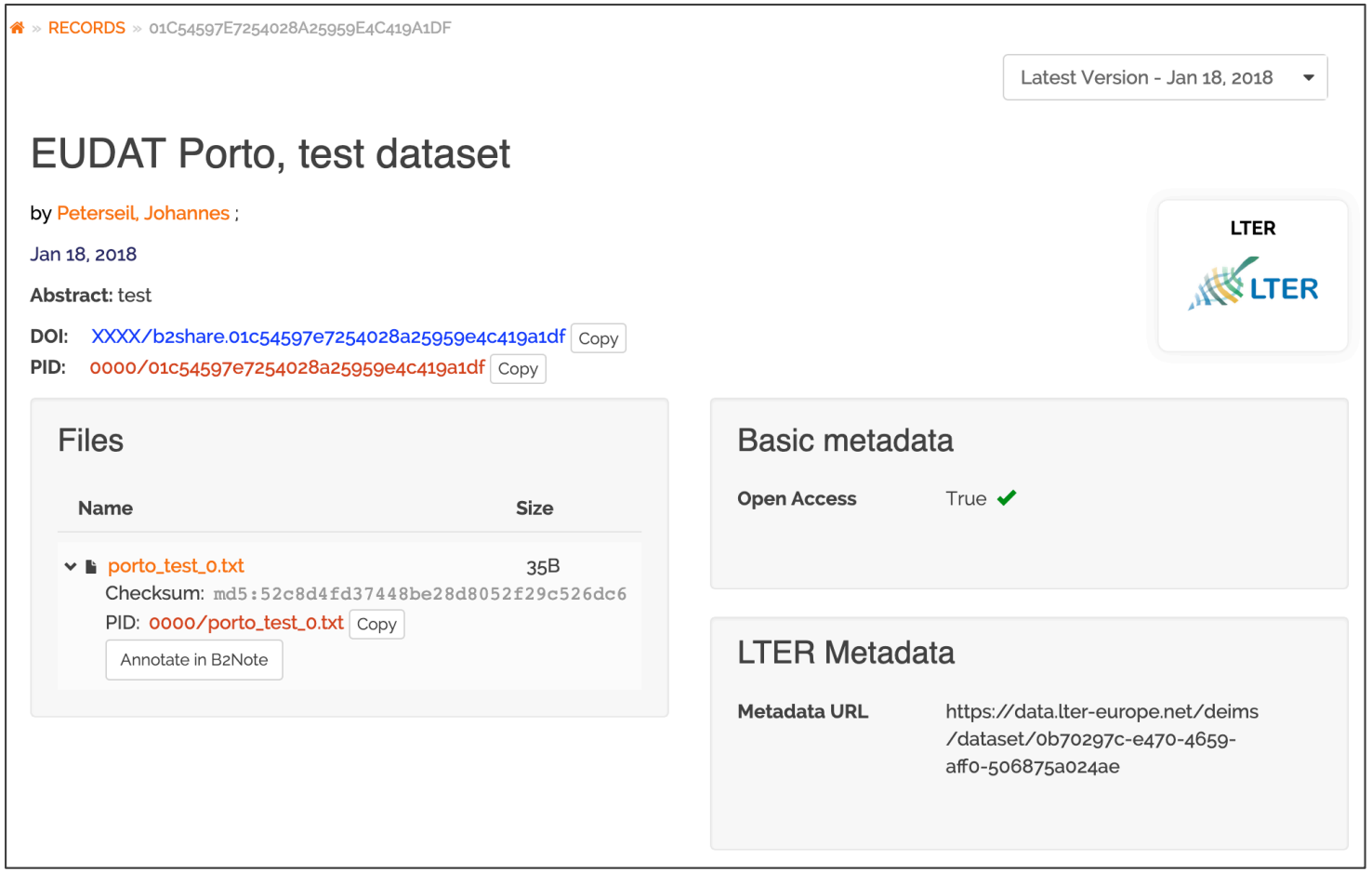
- UI-less interaction with B2SHARE!

Important concepts of B2SHARE

Records	• Data + metadata + state • Contained in a community
Metadata	• Default and community-specific • Governed by schemas
Communities	• Curation & review • Policies
Schemas	• Set of metadata fields • Enforce annotation

B2SHARE records

URL path



Metadata

Record PIDs

Files

Checksums / PIDs

Version

Community

Metadata

Community metadata

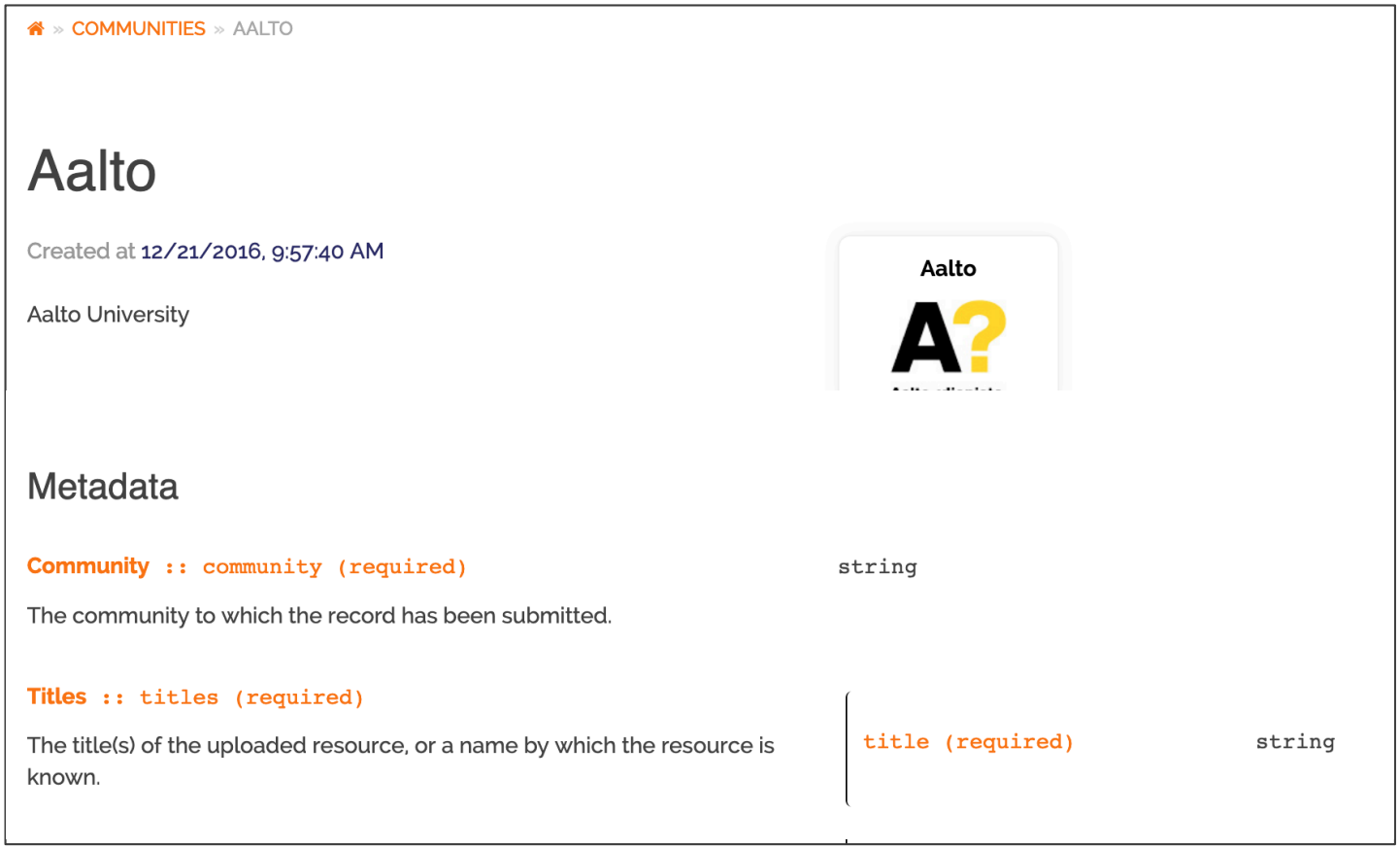
https://trng-b2share.eudat.eu/records/01c54597e7254028a25959e4c419a1df

B2SHARE communities

URL path

Metadata

Community metadata schema



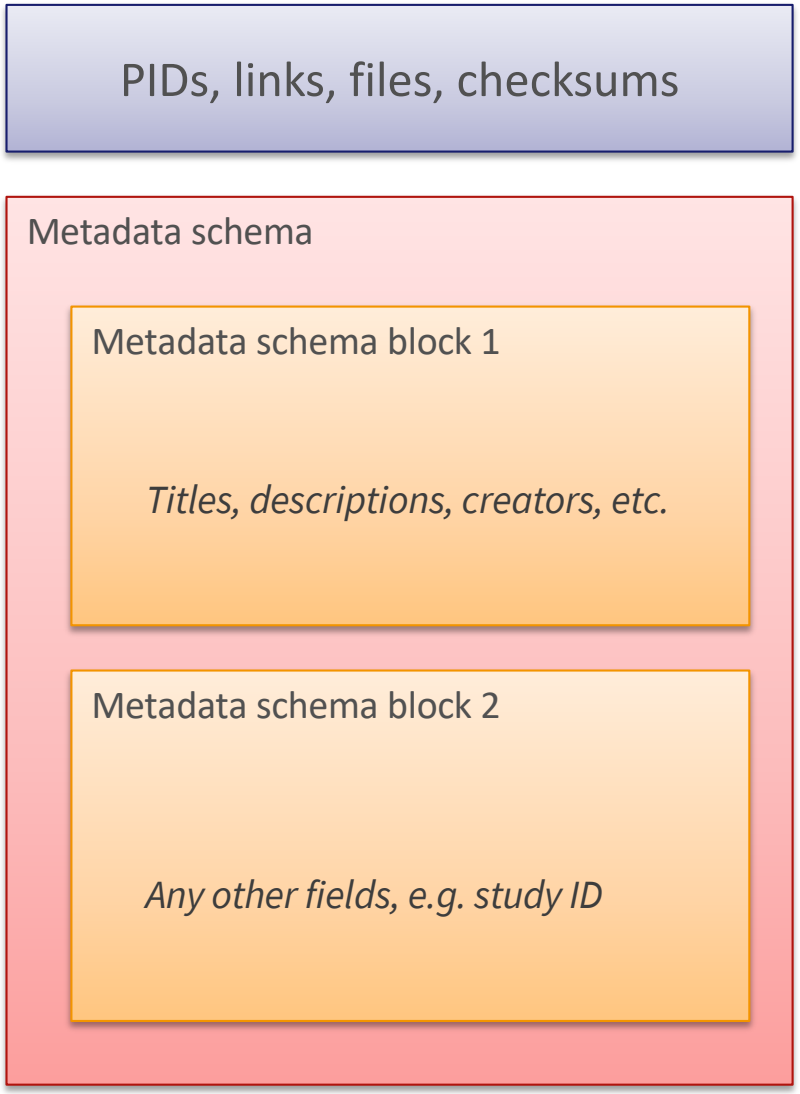
Logo

Not shown:
- Members
- Roles

https://trng-b2share.eudat.eu/communities/Aalto

Metadata

B2SHARE record metadata



- Technical metadata
- Default metadata (DataCite)
- Community metadata

Community metadata schemas

- Every community defines its **own metadata schema**
  - Defined using **JSON Schema**
  - Field definitions, vocabularies, etc.
  - May contain subfields and arrays
  - May contain community-specific fields
- Definitions are **publically** available, e.g. EUDAT:

```
"b2share": {
    "presentation": {
        "major": [
            "community",
            "titles",
            ...
            "contact_email"
```

```
    "required": [
        "community",
        "titles",
        "open_access",
        "publication_state",
        "community_specific"
    ],
```

Example metadata field definitions

- String field

```
▼ publication_date:
  ▼ description:                "The date when the data was or will be made
                                publicly available (e.g. 1971-07-13)"
    format:                     "date"
    title:                      "Publication Date"
    type:                       "string"
```

Example metadata field definitions

- Field using vocabulary

```
publication_state:
    description:                "State of the publication workflow."
    enum:
        0:                      "draft"
        1:                      "submitted"
        2:                      "published"
    title:                      "Publication State"
    type:                       "string"
```

Example metadata field definitions

- Field with subfields

```
▼ license:
    additionalProperties:          false
  ▼ description:                   "Specify the license under which this data set is
                                   available to the users (e.g. GPL, Apache v2 or
                                   Commercial). Please use the License Selector for
                                   help and additional information."
  ▼ properties:
    ▼ license:
        type:                      "string"
    ▼ license_uri:
        format:                    "uri"
        title:                     "URL"
        type:                      "string"
  ▼ required:
      0:                           "license"
    title:                         "License"
    type:                          "object"
```

Example metadata field definitions

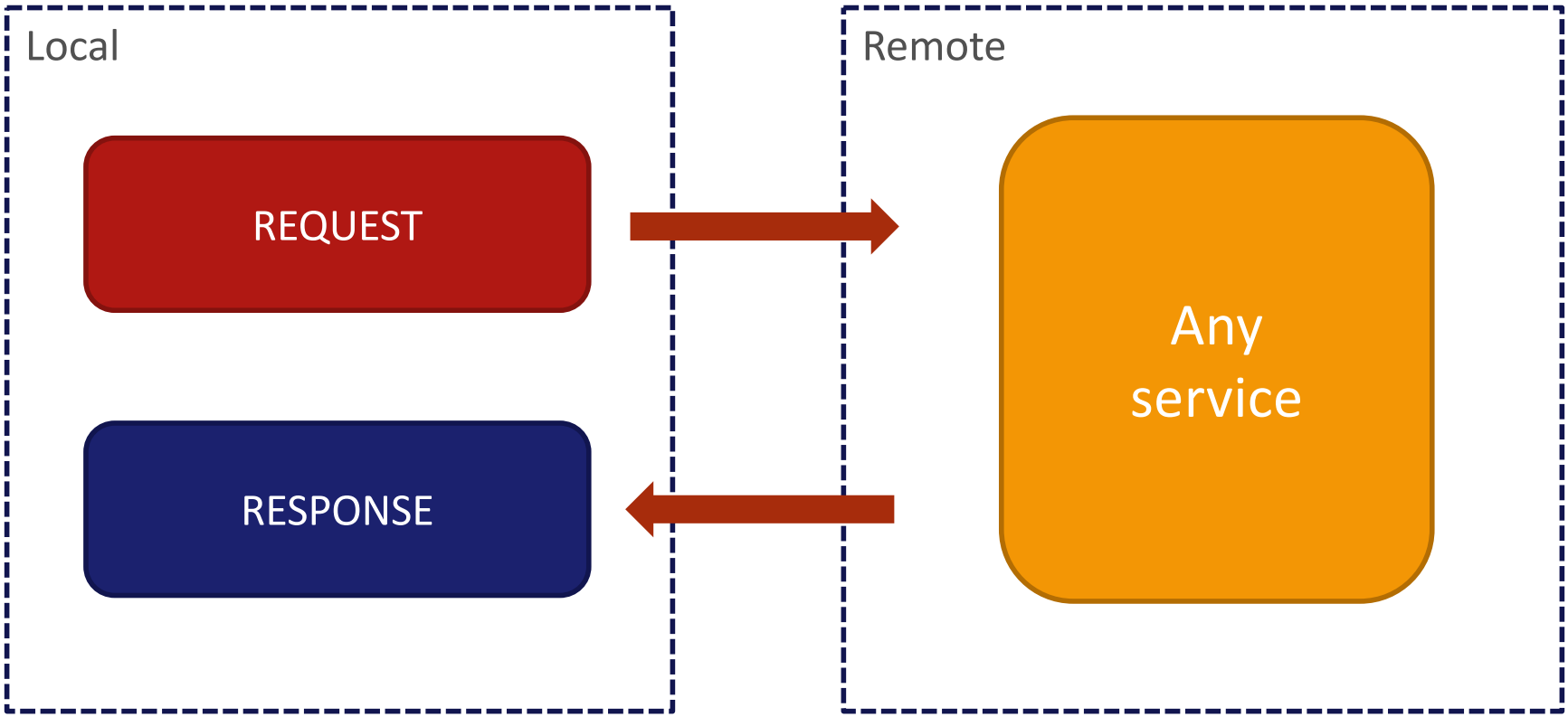
- Array field with subfields

```
creators:
  description:                  "The full name of the creators. The personal name
                                format should be: family, given (e.g.: Smith,
                                John)."
  items:
    additionalProperties:       false
    properties:
      creator_name:
        type:                   "string"
    required:
      0:                        "creator_name"
    type:                       "object"
  title:                        "Creators"
  type:                         "array"
  uniqueItems:                  true
```

Making requests

Making a request

- Use tool or application, or browser



Request structure

- Every request must have a **method** and **URL**
- Message containing:
  - Method, URL, protocol
  - Headers
  - Request body

```
GET /api/communities HTTP/1.1
Host: trng-b2share.eudat.eu
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.14; rv:69.0) Gecko/20100101 Firefox/69.0
Accept: application/json, text/javascript
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Referer: https://trng-b2share.eudat.eu/records/new
X-Requested-With: XMLHttpRequest
Content-Type: application/json
DNT: 1
Connection: keep-alive
Cookie: session=af166b37ec3c167e_5d84d17a.jpGSEZ_WKC_-_z1L2LyDkp1wLvQ
```

HTTP request method

- Different methods have different meaning, but up to service on how to process them

GET	• Requests data from a specified resource
POST	• Submits data to be processed to a specified resource
PUT	• Upload a representation of the specified URI, e.g. a file
PATCH	• Modify state of specified resource
DELETE	• Delete a specified resource

Request variables

- B2SHARE defines several **request variables** that function as identifiers for objects in B2SHARE

COMMUNITY_ID	• UUID or name • identifier of a community
RECORD_ID	• UUID • identifier for a specific record
FILE_BUCKET_ID	• UUID • identifier for a set of files of a specific record
FILE_NAME	• String • Name of a file contained in file bucket

Endpoints

- Every request is done using an endpoint:

What	Endpoint	Description
Search	`/api/records`	(Filtered) search for records
Records	`/api/records/RECORD_ID`	Metadata of records
Files	`/api/files/FILE_BUCKET_ID/FILE_NAME`	File download
Communities	`/api/communities`	Metadata of communities
Schemas	`/api/communities/COMMUNITY_ID/schemas/last`	Metadata schemas
Modify	`/api/records/RECORD_ID/draft`	Record modification

Response structure

- Every request returns a response with **status code**, **header** and message **body**, *even when an error occurred*
- Message containing:
  - Protocol, status code, reason
  - Header lines
  - Empty line
  - Response body

HTTP/1.1 200 OK
Server: nginx/1.11.8
Date: Tue, 24 Sep 2019 07:57:40 GMT
Content-Type: application/json
Content-Length: 12092
Connection: keep-alive
Set-Cookie: session=af166b37ec3c167e_5d84d17a.jpGSEZ_WKC_-_z1L2LyDkp1wLvQ; HttpOnly; Path=/api

HTTP responses

Informational	Success	Redirection	Client error	Server error
• 100: Continue • 101: Switch protocol • 102: Processing	• **200: OK** • **201: Created** • 202: Accepted • **204: No content** • 205: Reset content	• 300: Multiple choices • 301: Moved permanently • 302: Found • 303: See other • 304: Not modified	• **400: Bad request** • **401: Unauthorized** • 402: Payment required • **403: Forbidden** • **404: Not found**	• **500: Internal server error** • 501: Not implemented • 502: Bad gateway • 503: Service unavailable

Authentication through the API

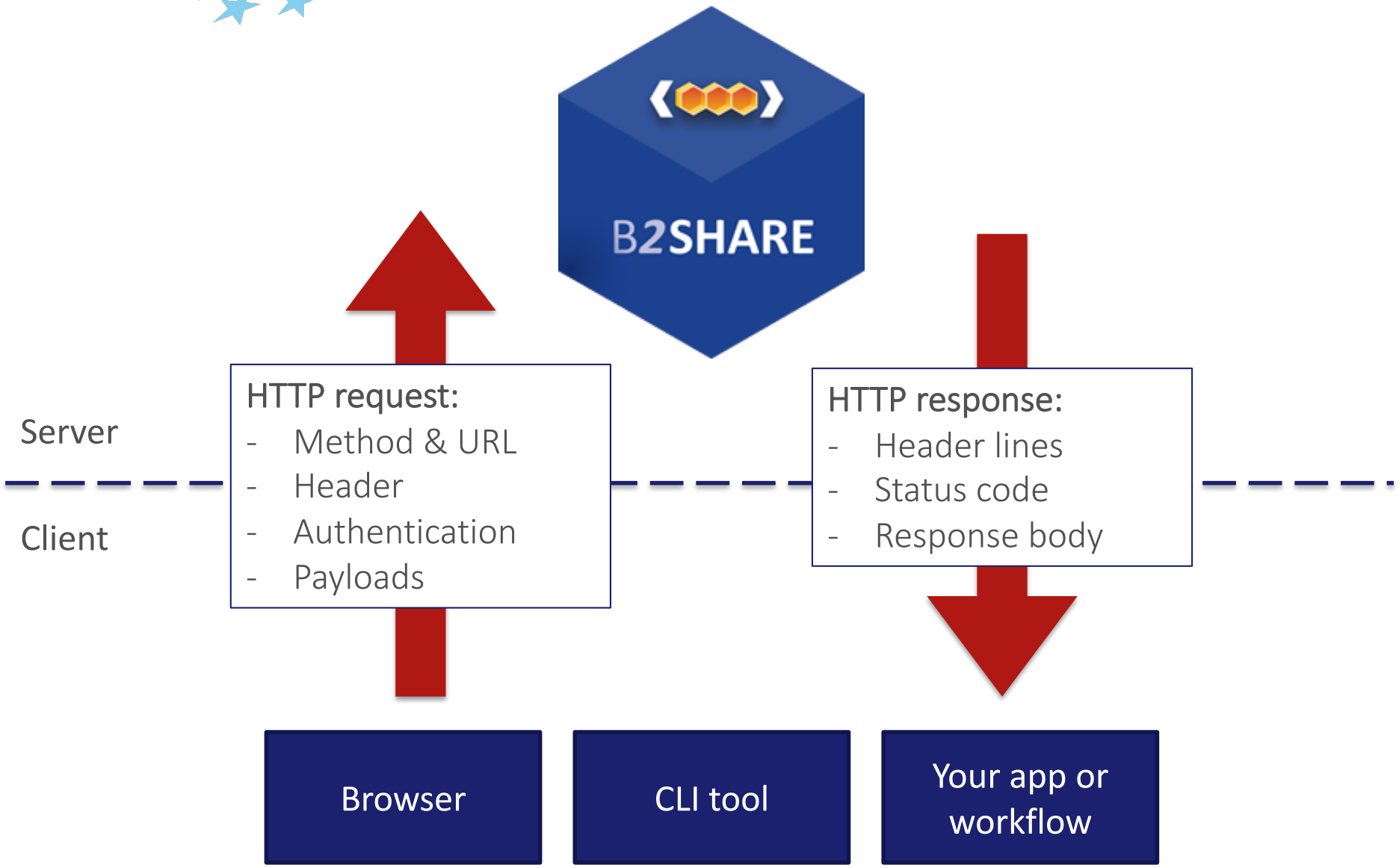
- B2SHARE does not accept username and password combination, instead use an **access token** for authentication!
  - Automatically generated unique string of characters attached to your account in B2SHARE
  - Only known by the owner, do not share with others!
  - Must be done in B2SHARE web interface

A new access token has just been created:

Name EUDAT Summer School

Access token pBI1bZgAB20A9nxqFl76c8zs78aAMFpeZjRemqpUfZ8aYs64JFv4V9OYhyE0

HTTP request overview



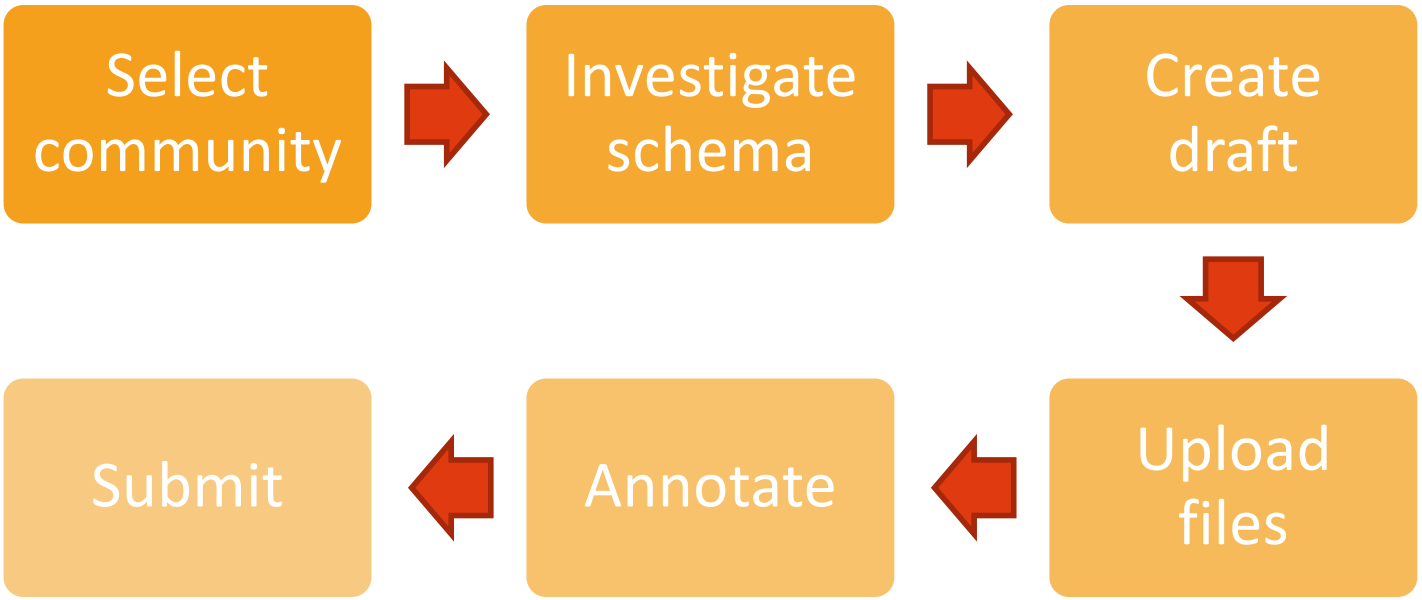
Publication workflow

B2SHARE publications

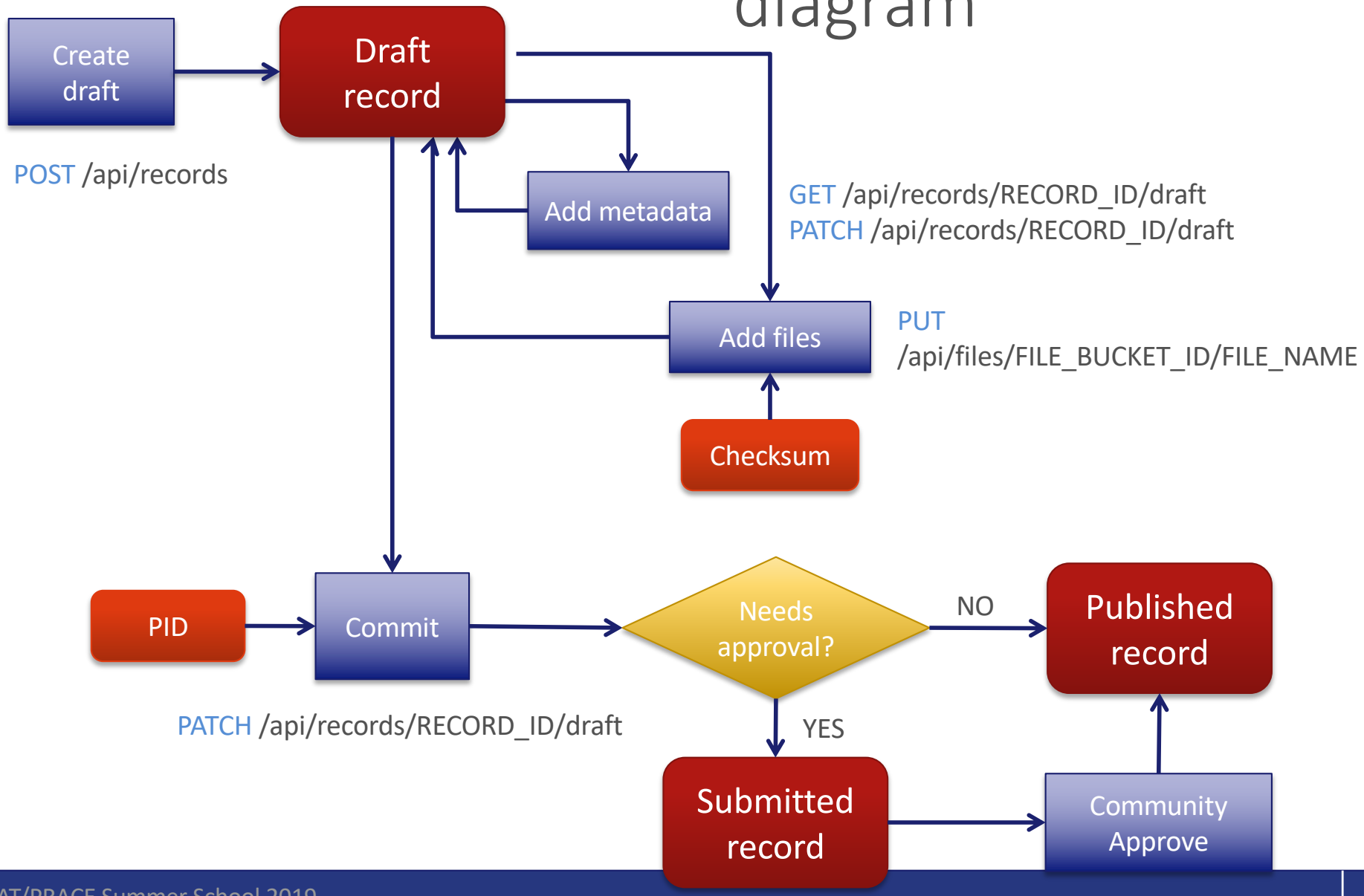
- B2SHARE publications:
  - As part of a selected community
  - Have at least one file
  - Mandatory metadata must be provided
- B2SHARE will provide:
  - Record with landing page
  - Unique URL
  - Persistent identifiers (Handle and DOI)
  - Harvesting by metadata catalogues

Full B2SHARE publication workflow

- Publishing in B2SHARE using the API involves **multiple steps**, therefore **multiple requests**

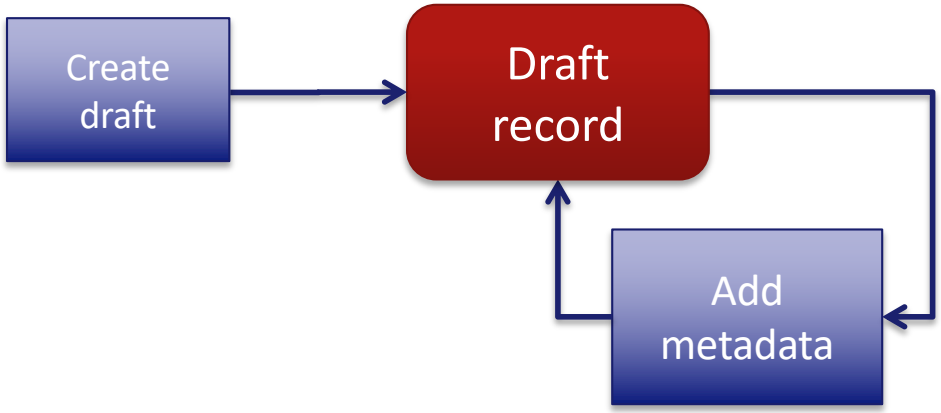


Full publication request diagram



Adding metadata

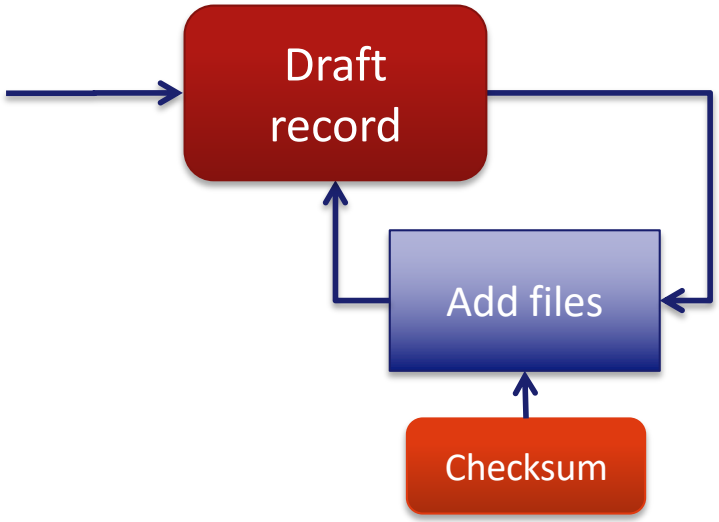
- Metadata must be added to a record using the API:
  - Upon creation of a draft record in a POST request with JSON data
  - Or by providing so-called **JSON patches** in a PATCH request



POST https://trng-b2share.eudat.eu/api/records/
PATCH https://trng-b2share.eudat.eu/api/records/RECORD_ID/draft

Adding files

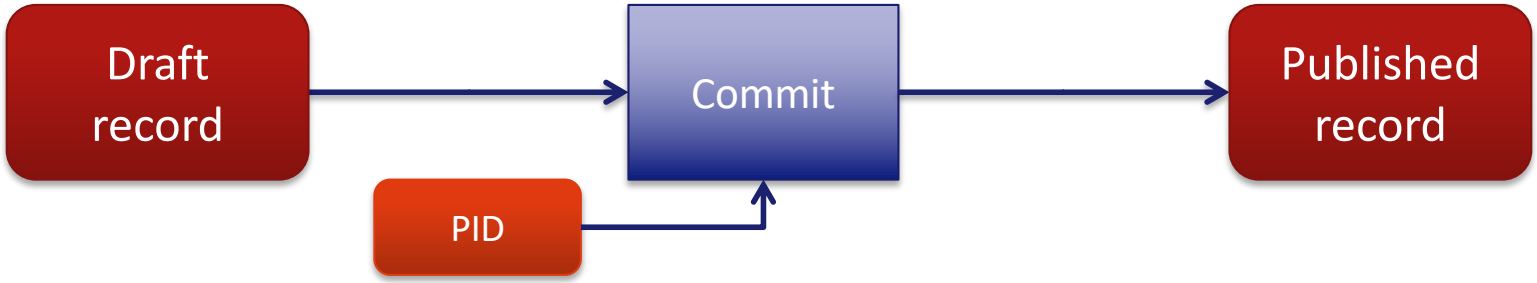
- Files can be added during the draft phase of your new record using a **PUT request**



PUT https://trng-b2share.eudat.eu/api/files/FILE_BUCKET_ID/FILE_NAME

Publishing your draft record

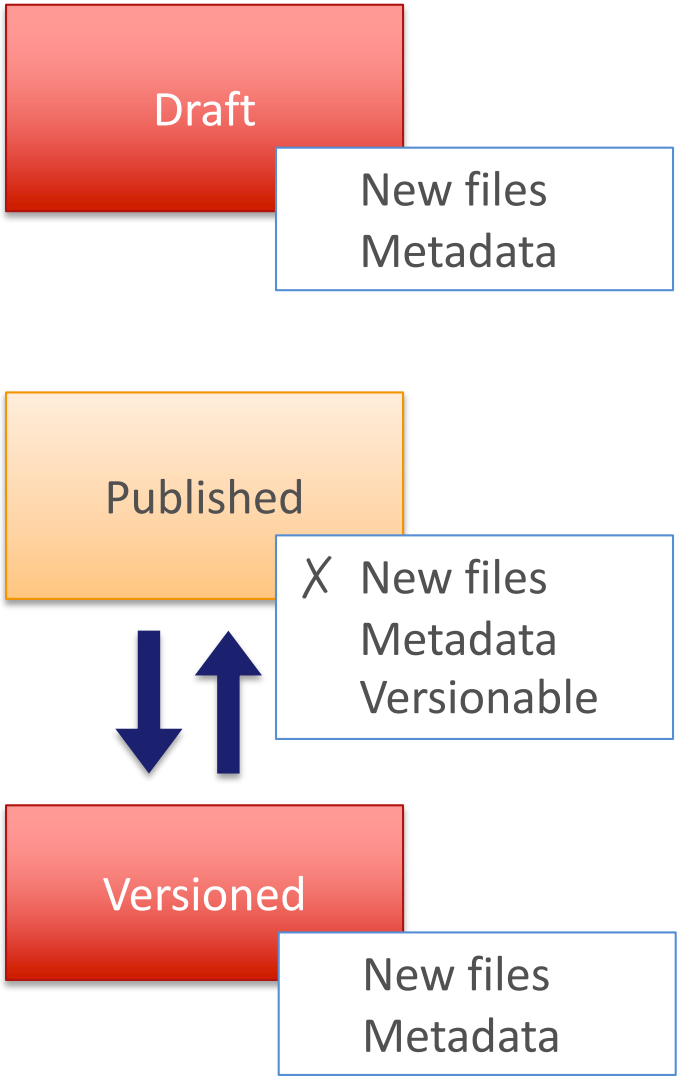
- Draft records are published by altering the value of the publication state in the metadata
- Once your record is published:
  - Files can not be changed or added anymore!
  - Metadata can be changed
  - **Persistent identifiers** are automatically added



PATCH https://trng-b2share.eudat.eu/api/records/RECORD_ID/draft

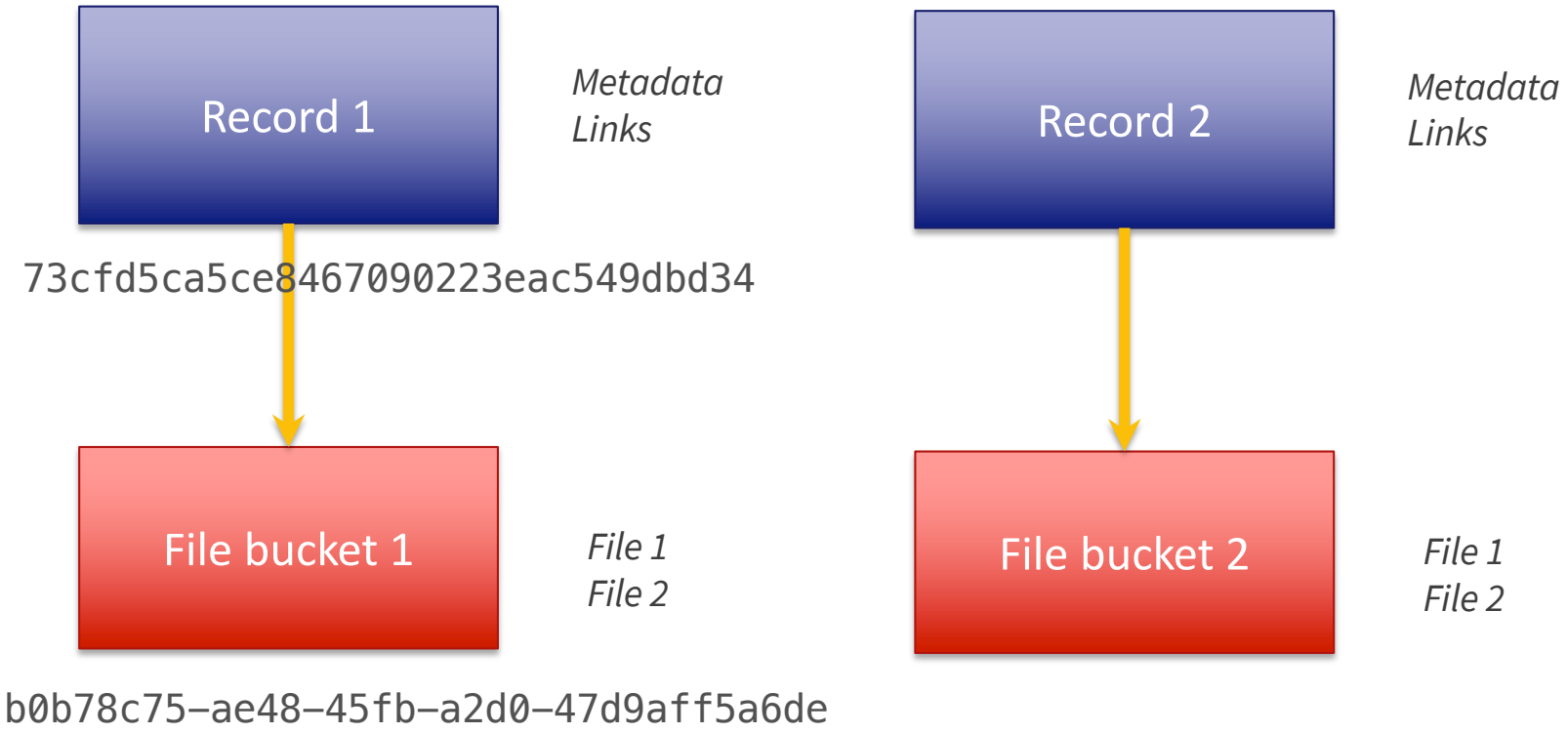
Draft records and versioning

- Draft records:
  - Updateable
  - State: 'draft'
- Published records:
  - Metadata updateable
- Record versioning:
  - Existing published records only
  - New PIDs!



Record file buckets

- Every record has its own file bucket containing the files



B2SHARE and Python

Simple examples

- Host: https://trng-b2share.eudat.eu
- HTTP method: **GET**
- Retrieve all existing records:

```
r = requests.get('https://trng-b2share.eudat.eu/api/records')
```

- List all communities:

```
r = requests.get('https://trng-b2share.eudat.eu/api/communities')
```

- Search for specific records of a community:

```
r = requests.get('https://trng-b2share.eudat.eu/api/records?    \
        q=community:COMMUNITY_ID`)
```

Complex example

- HTTP method: **POST**
- Create draft record ‘My test upload’:

```
parameters = {'access_token': token}
```

```
header = {‘Content-Type’: 'application/json'}
```

```
metadata = {"titles": [{"title":"My test upload"}],
                "community": "e9b9792e-79fb-4b07-b6b4-b9c2bd06d095",
                "open_access": True}
```

```
r = requests.post('https://trng-b2share.eudat.eu/api/records/',
        params=parameters,
        data=json.dumps(metadata),
        headers=header)
```

JSON Patch

- A set of **operations** that alter an existing set of metadata fields based on another set of fields

```
[
    {
        "path": "/community_specific",
        "value": {},
        "op": "add"
    },
    {
        "path": "/disciplines",
        "value": [
            "EUDAT Summer School"
        ],
        "op": "add"
    }
]
```

=

Metadata PATCH

Hands-on exercises

Today's hands-on assignment

- Publish a new record in B2SHARE
  - Using direct API requests only
  - Using Python Jupyter notebook

- Regarding B2SHARE:
  - Make sure to use the **training instance** of B2SHARE for your publications!
  - Use the **training instance** to generate an API token
  - Every student uses his/her own B2SHARE account

- Download the Jupyter notebook to your Galileo account

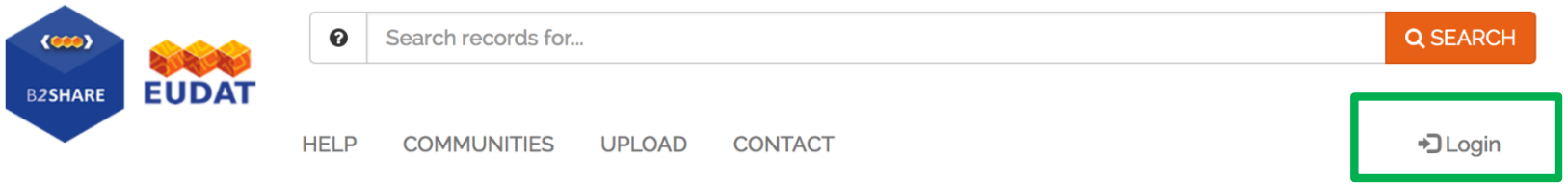
General instructions

- Create an **API token** on the B2SHARE training website
- Requirements for each request:
  - Request **URL** and **HTTP method** (e.g. GET, PUT)
  - Optional:
    - Object identifiers (e.g. record, community)
    - Additional parameters (e.g. your token)
    - Data payloads (e.g. files or metadata)
- B2SHARE API endpoint: /api

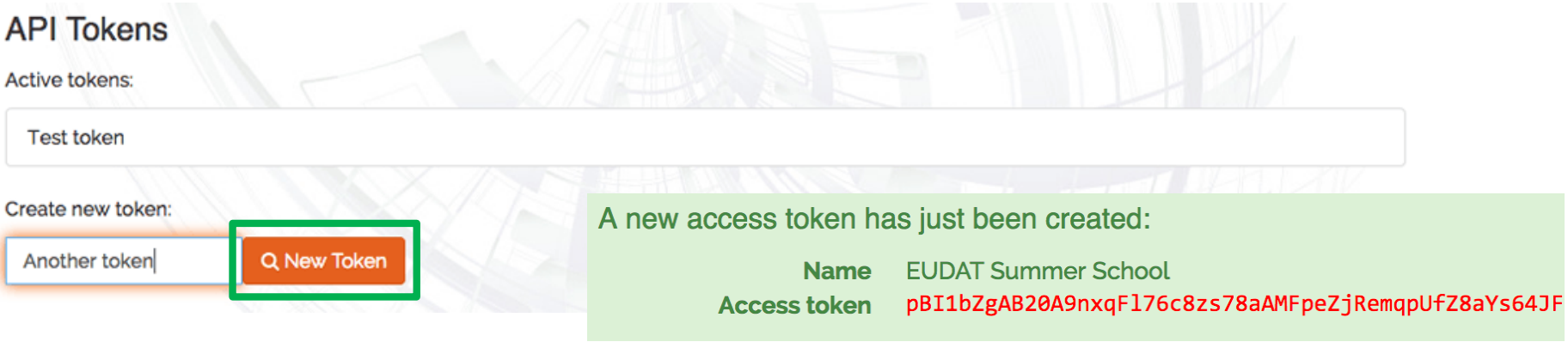
- Open Jupyter notebook and work through it!

Getting your access token

- Log in on B2SHARE and navigate to profile page:



- Create a token by entering a new name:



- Click on 'New token'

https://b2share.eudat.eu

For more info: https://eudat.eu/services/b2share

B2SHARE User Documentation:
https://eudat.eu/services/userdoc/b2share

B2SHARE Training presentations:
https://www.eudat.eu/b2share-training-suite

B2SHARE hands-on training:
https://github.com/EUDAT-Training/B2SHARE-Training

Thank you!

Authors

Hans van Piggelen, SURFsara

Contributors