LEIBNIZ-INFORMATIONSZENTRUM
TECHNIK UND NATURWISSENSCHAFTEN
UNIVERSITÄTSBIBLIOTHEK

TIB

Einführung in die DataCite REST API

Marleen Burger
Technische Informationsbibliothek (TIB)
marleen.burger@tib.eu

Leibniz Gemeinschaft

Agenda

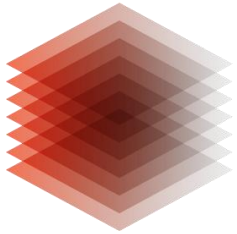
1. Allgemeines zu APIs

2. Abrufen von Inhalten
   I. Abrufen einzelner DOI
   II. Suchanfragen

3. Registrieren und Updaten
   I. Draft
   II. Findable
   III. Updaten

4. Weiterführende Informationen

LEIBNIZ-INFORMATIONSZENTRUM
TECHNIK UND NATURWISSENSCHAFTEN
UNIVERSITÄTSBIBLIOTHEK



TIB

1. Allgemeines zu APIs

Leibniz
Gemeinschaft

1. Was ist eine API?

- Abkürzung für **Application Programming Interface**, auf deutsch häufig **Programmierschnittstelle**
- Dient dem standardisierten Informationsaustausch zwischen einer Anwendung und anderen Programmen oder Programmteilen
- Definiert genau die Form von Verarbeitungsanfragen und Rückgaben
- „Maschine-Maschine-Kommunikation“

1. REST APIs

- RESTful, d. h. verwendet die REST-Architektur

- REST (Representational State Transfer) ist kein Standard, sondern die Orientierung an Richtlinien, die viele Paradigmen und quasi-Standards des WWWs vereinen (z. B. bezüglich des Client-Server-Austauschs)

- Nutzen **HTTP-Befehle**, um z. B. mittels GET, PUT, POST und DELETE auf Daten zuzugreifen:
  - GET – Ressource vom Server anfordern
  - POST – Anlegen einer neuen Ressource
  - PUT – Updaten einer Ressource
  - DELETE – Ressource löschen

1. DataCite REST API

- Ermöglicht das **Abrufen** von DataCite-DOI-Metadaten ohne Authorisierung
- **Registrieren** und **Updaten** von DOIs nach Authorisierung
- Nachfolger der MDS API
- Aktuelle Version 2
- RESTful
- Entspricht den JSONAPI-Spezifikationen und gibt Ergebnisse im **JSON-Format** aus

1. Wie spricht man eine API an?

- APIs verwenden **kein GUI** (Graphical User Interface), Befehle müssen daher genau und komplett sein

- Diverse Methoden der Ansprache, generell: muss **HTTP-Befehle verarbeiten** können. Infrage kommen daher z. B. verschiedene **Programmiersprachen**, wie JavaScript oder Python

- Der Abruf von REST-API-Inhalten ist aber auch über URL-Manipulation im Browser mittels **URL-Parametern** möglich. Für die Registrierung, das Updaten oder Datei-Uploads funktioniert dieses Vorgehen allerdings nicht

1. cURL

- **Einfaches Kommandozeilenprogramm inkl. Library zur Datenübertragung**
- Standardwerkzeug in der Softwareentwicklung und zur Verwendung von APIs
- Open Source
- In vielen Linuxdistributionen und Mac OS standardmäßig integriert, ab Windows 10 vorinstalliert

https://curl.se/

LEIBNIZ-INFORMATIONSZENTRUM
TECHNIK UND NATURWISSENSCHAFTEN
UNIVERSITÄTSBIBLIOTHEK

TIB

2. Abrufen von Inhalten

Leibniz
Gemeinschaft

2. Abrufen von Inhalten

Abruf über cURL:

Kommandozeile öffnen und den Befehl curl gefolgt von der URL angeben.
Speicherung des Ergebnisses in einer Datei ist möglich; die Datei wird unter dem spezifizierten Namen im aktuellen Verzeichnis abgelegt.

Abruf über URL-Parameter:

URL in der Adresszeile des Browsers angeben.
Speichern des Ergebnisses ist durch Kopieren und Einfügen in einen beliebigen Texteditor möglich.

2. Abrufen von Inhalten

URL-Parameter

- ermöglichen die Verwendung **komplexer Suchanfragen**
- Bestehen aus einem **Variable-Value-Paar**, das an die URL angehängt wird
- Trennzeichen zwischen URL und URL-Parametern ist ein Fragezeichen **?**
- Trennzeichen zwischen verschiedenen URL-Parametern ist das Ampersand **&**
- Values werden mit einem Gleichheitszeichen angegeben **=**
- Bestimmte Zeichen müssen **codiert** werden, z. B. wird ein Leerzeichen als %20 angegeben

2.I Abrufen einzelner DOIs

cURL:
$ curl https://api.datacite.org/dois/10.5446/31473

$ // Verzeichnis oder directory, z. B. C:\Users\Username\Desktop
curl // Kennzeichnung eines cURL-Kommandos

cURL mit Speichern in einer Datei:
$ curl https://api.datacite.org/dois/10.5446/31473 **> doi.json**

> **doi.json** // legt Ergebnis in einer Datei mit hier festgelegtem Namen und festgelegter Dateiendung im aktuellen Verzeichnis ab

URL:
https://api.datacite.org/dois/10.5446/31473

2.II Suchanfragen - Queries

Queries durchsuchen standardmäßig alle Metadatenfelder, sie können aber auf bestimmte Felder beschränkt werden

Beispiele:

1 - Stichwortsuche in sämtlichen Metadatenfeldern nach „Tree Frog“
https://api.datacite.org/dois?query=tree+frog

2 - Liste aller DOIs, die ein mit dem Objekt in Beziehung stehendes Objekt angegeben haben
https://api.datacite.org/dois?query=relatedIdentifiers:*

3 - Liste aller DOIs, die eine Zitation mittels „Cites“ angegeben haben
https://api.datacite.org/dois?query=relatedIdentifiers.relationType:Cites

Kombination aus Beispiel 1 & 2
https://api.datacite.org/dois?query=relatedIdentifiers:*+tree+frog

2.II Suchanfragen - Filter

Filter sind von der API vordefinierte Listen, die nicht über dasselbe Befehlsformat wie Queries erzeugt werden:

Liste aller DOIs, die den ResourceTypeGeneral „Text“haben
https://api.datacite.org/dois?resource-type-id=text

Liste aller DOIs eines Repositories
https://api.datacite.org/dois?client-id=tib.kmo

Kombination mit Queries möglich:
https://api.datacite.org/dois?query=resource-type-id=text+tree+frog

Weitere ausgewählte Filter:

provider-id	DataCite Provider (z. B. TIB)
person-id	Mit einer bestimmten ORCiD verknüpfte Metadaten
created	Erstellungsdatum einer DOI
schema-version	Metadatenschema-Version

2.II Suchanfragen

Weitere Modifikatoren

Anzahl der Einträge in einer Liste pro Seite auf 10 setzen (Standard 25)
https://api.test.datacite.org/dois?query=tree+frog&page[size]=10

Auf Seite 4 in der Liste der Einträge springen
https://api.test.datacite.org/dois?query=tree+frog&page[number]=4

2.II Suchanfragen

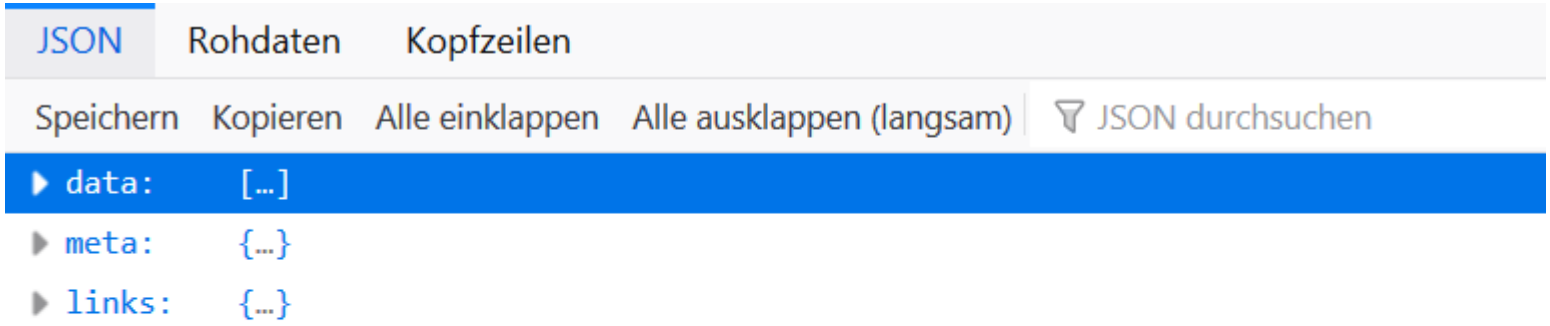
Queries und Filter werden analog zu einzelnen DOIs abgerufen

cURL:
$ curl https://api.datacite.org/dois?query=relatedIdentifiers:*+tree+frog

cURL mit Speichern in einer Datei:
$ curl https://api.datacite.org/dois?query=relatedIdentifiers:*+tree+frog **>**
doiliste.json

URL:
https://api.datacite.org/dois?query=relatedIdentifiers:*+tree+frog

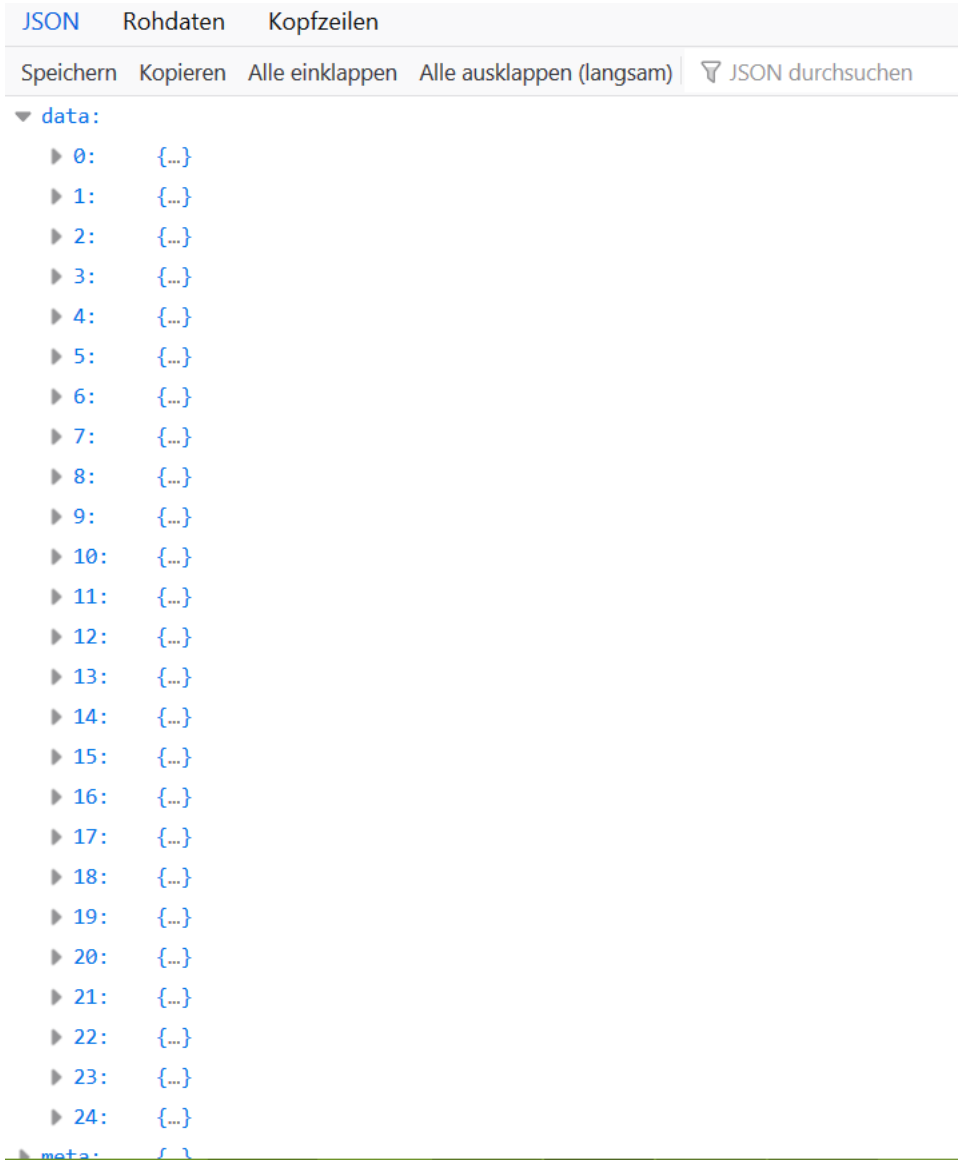
2.II Suchanfragen – Ergebnisliste im Browser



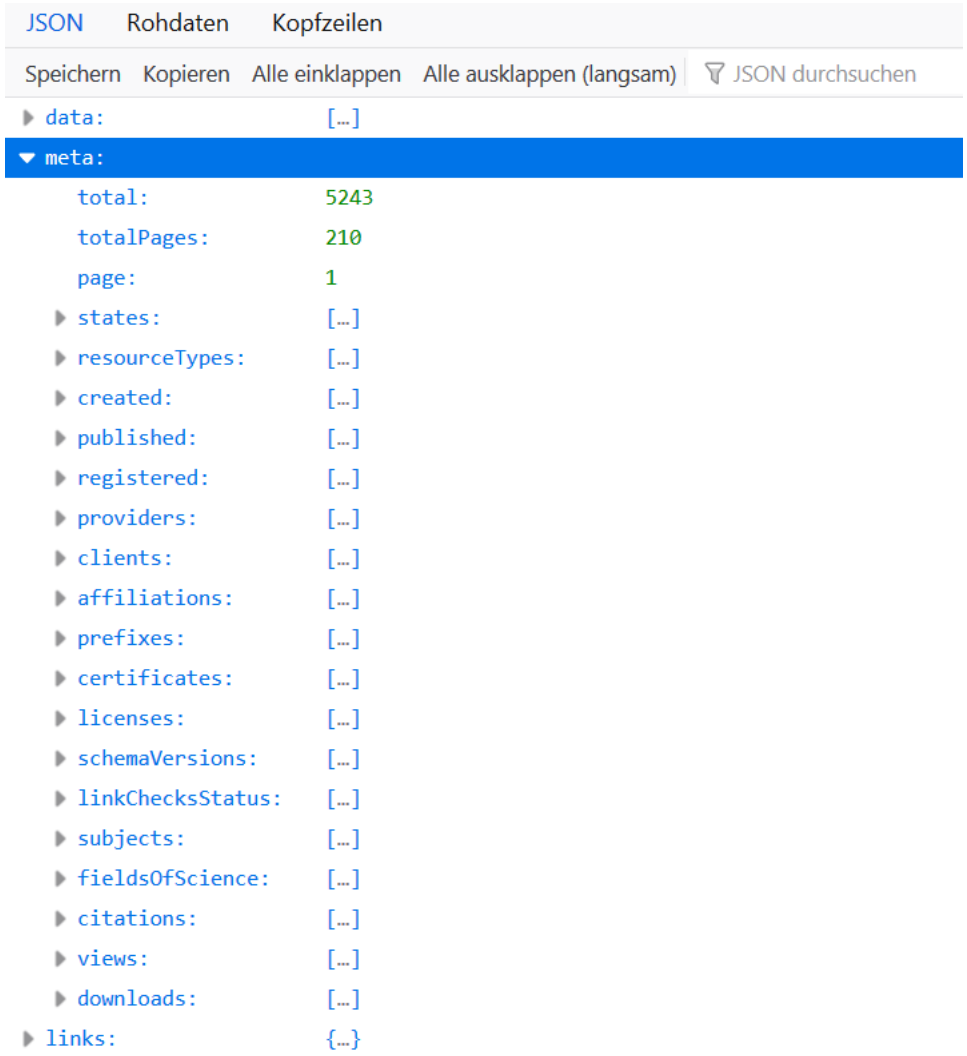
Data – enthält die Datensätze, die den Kriterien der Suche entsprechen
Meta – Informationen über das Ergebnis der Suche, z. B. wie viele Datensätze gefunden wurden
Links – enthält Links zur aktuellen und zur nächsten Seite

Datenreiter werden mit Klick auf die Pfeile ein- und ausgeklappt

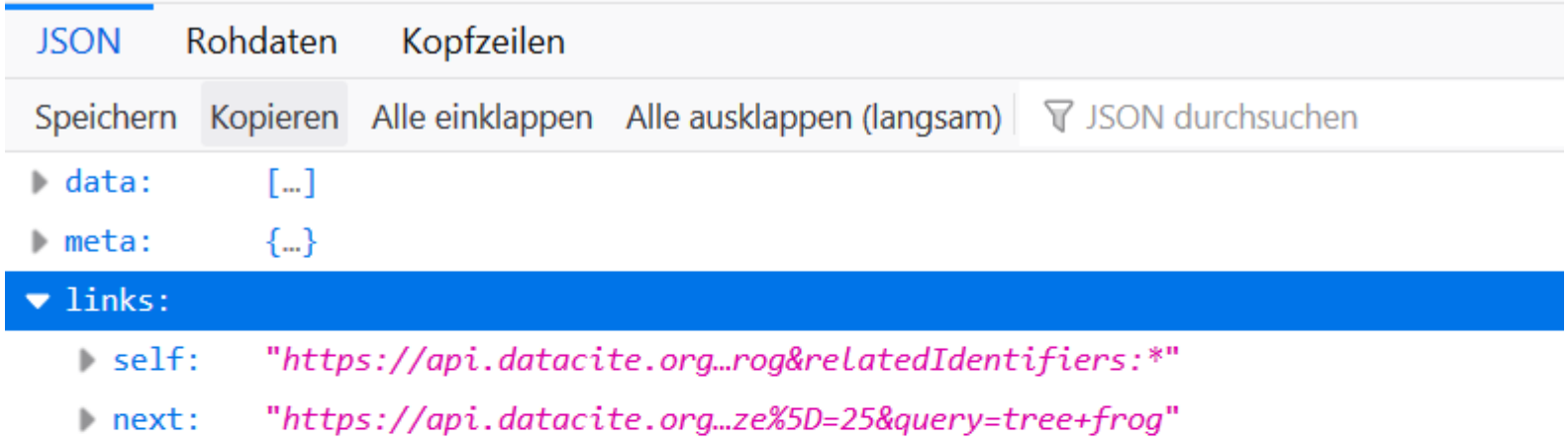
2.II Suchanfragen – Ergebnisliste im Browser



2.II Suchanfragen – Ergebnisliste im Browser



2.II Suchanfragen – Ergebnisliste im Browser



ACHTUNG:
Wenn man den Abruf über den Browser startet, werden fehlerhaft angegebene Parameter nicht als solche angezeigt, sondern aus der Suchanfrage abgeschnitten.
Im Bereich Link der erzeugten Daten wird der tatsächlich ausgeführte Befehl angezeigt.

LEIBNIZ-INFORMATIONSZENTRUM
TECHNIK UND NATURWISSENSCHAFTEN
UNIVERSITÄTSBIBLIOTHEK

TIB

3. Registrieren und Updaten

Leibniz
Gemeinschaft

3.I - Registrieren eines Draft DOIs

JSON-Datei (mit für die Registrierung nötigen minimalen Metadaten) erstellen (z. B. in Notepad ++):

```
{
  "data": {
    "type": "dois",
    "attributes": {
      "doi": "10.80856/o6ksb"
    }
  }
}
```

Abspeichern als .json unter beliebigem Namen (hier im Beispiel minimale_metadaten_draft.json)

3.I - Registrieren eines Draft DOIs

Alternative für die Generierung eines zufälligen Suffixes:

```
{
  "data": {
    "type": "dois",
    "attributes": {
      “prefix": "10.80856"
    }
  }
}
```

3.I - Registrieren eines Draft DOIs

Mit cURL folgenden Befehl ausführen:

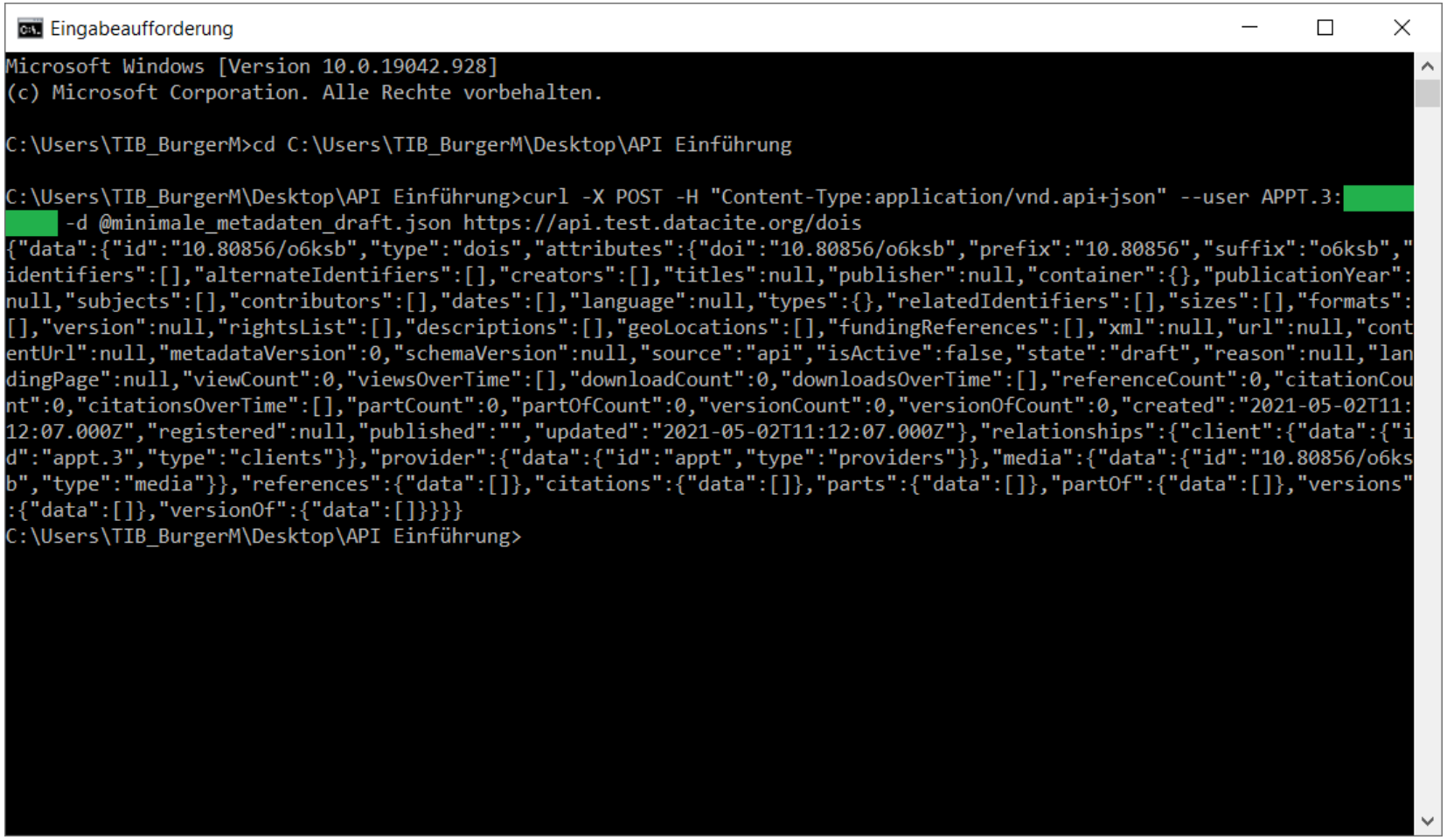
$ curl -X POST -H "Content-Type:application/vnd.api+json" --user **REPOSITORY-ID**:**Passwort** -d @**minimale_metadaten_draft.json** https://api.test.datacite.org/dois

3.I - Registrieren eines Draft DOIs

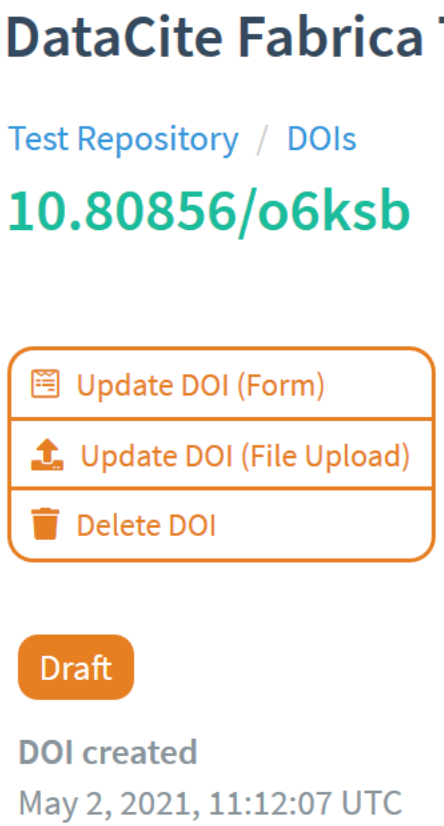
$ // directory, z. B. C:\Users\Username\Desktop
curl // Kennzeichnung eines cURL-Kommandos
-X POST // Führt den HTTP-Befehl POST aus
-H "Content-Type:application/vnd.api+json" // Spezifiziert Input-Format
--user **REPOSITORY-ID**:**Passwort** // Authorisierung
-d @**minimale_metadaten_draft.json** // Payload-Zieldatei
https://api.test.datacite.org/dois // API-Endpunkt

Wenn als Payload eine Datei angegeben wird, ist es wichtig, dass der Befehl aus demselben Directory ausgeführt wird, in dem die Datei liegt.
Andernfalls kann sie nicht gefunden und damit nicht gelesen werden.

3.I - Registrieren eines Draft DOIs



3.I - Registrieren eines Draft DOIs



Metadaten könnten über API-Update, Fabrica-Formular oder Fabrica-File-Upload ergänzt werden

3.II - Registrieren eines Findable DOIs

JSON-Datei erstellen:

```
{
 "data": {
   "id": "10.80856/zph6h",
   "type": "dois",
   "attributes": {
    "event": "publish",
    "doi": "10.80856/zph6h",
    "creators": [{
      "name": "Hofmann, Sigurd"
    }],
    "titles": [{
      "title": "Wie entdeckt man neue Elemente?"
    }],
    "publisher": "Beilstein-Institut zur Förderung der Chemischen Wissenschaften",
    "publicationYear": 2016,
    "types": {
      "resourceTypeGeneral": "Audiovisual"
    },
    "url": "https://av.tib.eu/media/50317",
    "schemaVersion": "http://datacite.org/schema/kernel-4"
   }
 }
}
```

3.II - Registrieren eines Findable DOIs

Mit cURL folgenden Befehl ausführen:

$ curl -X POST -H "Content-Type:application/vnd.api+json" --user **REPOSITORY-ID**:**Passwort** -d @**minimale_metadaten_findable.json** https://api.test.datacite.org/dois

Der Befehl ist identisch zu dem Registrieren eines Draft-DOIs, das Befüllen des attributes-Feldes „event“ mit „publish“ führt zur Registrierung eines DOIs im „Findable“-Status.

3.III – Update eines DOIs

Hinzufügen neuer Metadatenfelder und Update bereits befüllter Felder

Beispiele:

- Hinzufügen einer Zitation („relatedIdentifiers.Cites“)
- Änderung des Wertes im Feld resourceTypGeneral (von beliebigem Wert in „JournalArticle“)

3.III – Update eines DOIs (Hinzufügen)

Hinzufügen einer Zitation
JSON-Datei erstellen:

```
{
    "data": {
        "attributes": {
            "relatedIdentifiers": [
                {
                    "relatedIdentifier": "https://doi.org/10.80856/5bc9q",
                    "relatedIdentifierType": "DOI",
                    "relationType": "Cites",
                    "resourceTypeGeneral": "Dataset"
                }
            ]
        }
    }
}
```

3.III – Update eines DOIs (Hinzufügen)

Mit cURL folgenden Befehl ausführen:

curl -X PUT -H "Content-Type: application/vnd.api+json" --user **REPOSITORY-ID:Passwort** -d @**update_cites.json** https://api.test.datacite.org/dois/10.80856/zph6

Updates verwenden PUT, nicht POST!

3.III – Update eines DOIs (Änderung)

Änderung des Wertes von resourceTypeGeneral

JSON-Datei erstellen:

```
{
  "data": {
    "attributes": {
      "types": {
        "resourceTypeGeneral": „JournalArticle"
      }
    }
  }
}
```

3.III – Update eines DOIs (Änderung)

Mit cURL folgenden Befehl ausführen:

curl -X PUT -H "Content-Type: application/vnd.api+json" --user **REPOSITORY-ID:Passwort** -d @**update_resourcetypegeneral.json** https://api.test.datacite.org/dois/10.80856/zph6

3.III – Updaten mehrerer DOIs

JSON-Dateien zum Update enthalten keine Referenz auf Ihr Ziel, d. h. den DOI, auf den die Änderungen angewandt werden

--> diese Referenz ist im cURL-Befehl enthalten:

curl -X PUT -H "Content-Type: application/vnd.api+json" --user REPOSITORY-ID:Passwort -d @update_cites.json
https://api.test.datacite.org/dois/10.80856/zph6h

Für Batch-Updates kann daher mit einfachen Mitteln (z. B. in einem Tabellenkalkulationsprogramm) eine DOI-Liste in eine cURL-Befehlsliste übertragen werden, die dann das Updaten mehrerer DOIs mit minimalem Zeitaufwand ermöglicht.

4. Weiterführende Informationen

DataCite REST API Dokumentation (englisch):
https://support.datacite.org/docs/api

DataCite REST API Referenz (englisch):
https://support.datacite.org/reference/introduction

JSONAPI Spezifikation:
https://jsonapi.org/

RESTful API Tutorial
https://restfulapi.net/