SCOAP3 Technical Webinar

10TH & 17TH APRIL 2015

Table of content

How to start using API?

GENERAL CONCEPT

REGISTRATION

QUERY STRUCTURE

PERFORMING A REQUEST

General concept of API

Additional infrastructure

Unlimited access

Support

Training materials

Registration

- New user can register to the API visiting SCOAP3 API website and filling the registration form or clicking the button bellow.

api.scoap3.org/register

Query structure

Search pattern
- search pattern and other parameters
- URL-encoded

Base URL
- http://api.scoap3.org/search

API key
- apikey=xxxxx

Signature
- HMAC-SHA1algorithm
- you can use this website: **www.freeformatter.com/hmac-generator.html**
- signature=yyyyyy

Query structure – full query

Our search pattern	•**p=country:Switzerland**
URL-encodeing	•p=country**%3A**Switzerland
Adding base URL	•**http://api.scoap3.org/search?**p=country%3ASwitzerland
Adding API key	•http://api.scoap3.org/search?**apikey=XXXXXX**&p=country%3ASwitzerland – **alphabetical order**
Adding a signature	•Create signature for whole URL above •http://api.scoap3.org/search?apikey=XXXXXX&p=country%3ASwitzerland&**signature=YYYYY**
Run	•Perform a request using WGET, CURL, any programming language or web browser

Perform request

Different methods using HTTP GET:

- Web browser – easy to use but browser might crush while trying to render a very big XML file
- Wget
- cURL
- Any programming language (example in Python)

Perform request - Wget

„GNU Wget is a free software package for retrieving files using HTTP, HTTPS and FTP, the most widely-used Internet protocols. It is a non-interactive commandline tool, so it may easily be called from scripts, cron jobs, terminals without X-Windows support, etc."

```
user:~$ wget
    -O api_download.xml
     "http://api.scoap3.org/search?apikey=XXXXXX&p=country%3ASwitzerland&signature=YYYYY"
```

A downloaded XML will be saved in `api_download.xml` file.

Perform request - cURL

„cURL is a command line tool and library for transferring data with URL syntax, supporting DICT, FILE, FTP, FTPS, Gopher, HTTP, HTTPS, IMAP, IMAPS, LDAP, LDAPS, POP3, POP3S, RTMP, RTSP, SCP, SFTP, SMB, SMTP, SMTPS, Telnet and TFTP"

```
user:~$ curl
    -G                                                                  - send data with HTTP GET
    -o out.txt                                                          - set an output file
    --data "apikey=936e3e0e-eada-4f2d-b733-97027f822a69"                - HTTP data
    --data-urlencode "p=affiliation:Cambridge AND NOT country:USA"      - HTTP data URL-encoded
    --data "signature=438f50104f12f299a432a1c1c3af4c1cf52c16b4"          - HTTP data
    -v                                                                  - verbose (more output)
    "http://api.scoap3.org/search"                                      - base URL address
```

Perform request - Python

```
from urllib import urlopen

f = open("scoap3_download.xml", "w")
connector = urlopen("http://api.scoap3.org/search?apikey=xxx&p=affiliation%3ageneva&signature=yyy")
f.writelines(connector.readlines())
f.close()
```

This code snippet:

- creates a file „f" on your hard drive
- connects to SCOAP3 API server
- writes data from server to file
- Closes „f" file

Typical searches via API

- BASIC PARAMETERS
- VIRTUAL FIELDS
- TIME SPANS
- LOGICAL OPERATORS

Basic parameteres & virtual fields

p - pattern	of – output format	cc - collection	sf – sort field	so – sort order	ot – output tags
author:	hm	-- collection name --	-- as in pattern --	a – ascending	-- marc tags --
title:	hx			d - descending	
affiliation:	t				
doi:	ot				
orcid:	xm				
country:					
datecreated:					
-- marc tags --					

http://scoap3.org/scoap3-repository/xml-api - documentation and examples

Logical operators

Search supports logical operators like:

- AND
- AND NOT
- OR

They can be used to connect different virtual fields in „p” parameter

Time span

To determine a time span for records you are looking for use: datecreted

- p=datecreated:2014-01-08
- p=datecreated:2014-01-08->2014-01-31
- p=datecreated:2014-01-10:12:00:00->2014-01-10:15:00:00
- p=datecreated:2014-01-01->9999-01-01

Examples

- p=affiliation:geneva AND author:smith

https://repo.scoap3.org/search?p=affiliation%3Ageneva+and+author%3Asmith

- p=affiliation:cambridge AND NOT country:USA

http://repo.scoap3.org/search?p=affiliation%3Acambridge+and+not+country%3AUSA

- p=affiliation:cambridge AND NOT country:USA, ot=100,700, of=t

http://repo.scoap3.org/search?p=affiliation%3Acambridge+and+not+country%3AUSA&of=t&ot=100,700

- Cc=Acta, ot=520, of=t, sf=520__a

http://repo.scoap3.org/search?cc=Acta&of=t&ot=520__a&sf=520__a

Examples

- p=datecreated:2015-01-01->2015:12:31

http://repo.scoap3.org/search?p=datecreated%3a2015-01-01->2015%3a12%3a31

- p=orcid:0000-0001-5000-2897

http://repo.scoap3.org/search?p=orcid%3a0000-0001-5000-2897

Regular expressions

- SCOAP3 patterns support also regular expressions with MySQL syntax
- Documentation: https://dev.mysql.com/doc/refman/5.7/en/regexp.html

- p=affiliation:geneva or affiliation:cern = p=affiliation:/geneva|cern/

- Typical regular expression patterns that can be used are:
  - . – any character
  - *, + - Zero or more instances of preceding element, One or more instances of preceding element
  - ^, $ - Beginning of string, End of string

Searching for papers from university/ institution

MULTIPLE AFFILIATIONS

Multiple affiliation serach with regular expression

Simple example for Jagiellonian University, Poland:

- p=affiliation:UJ or affiliation:Jagiellonian or affiliation:Jagiellonski
- p=affiliation:/uj | jagiellonian | jagiellonski/

Complicated example for search of The US Department of Energy (DOE) associated institutions:

- p=affiliation:/ames lab | argonne nat | brookhaven nat | bettis atom | fermi nat | idaho nat | jefferson nat | johns hopkins univ appl | kansas city plant | knolls atom | lawrence berkeley nat | lawrence livermore nat | los alamos nat | nat energy technol | nat renew energy | nevada nat | oak ridge nat | pacific northwest nat | pantex | princeton plasma | sandia nat | savannah river nat | slac | stanford linear/

How to download fulltext files?

FULLTEXT

METADATA SCHEMA

Files in metadata schema

```
<datafield tag="856" ind1="4" ind2=" ">
  <subfield code="s">223272</subfield>
  <subfield code="u">http://repo.scoap3.org/record/9789/files/fulltext.pdf</subfield>
</datafield>
<datafield tag="856" ind1="4" ind2=" ">
  <subfield code="s">69426</subfield>
  <subfield code="u">http://repo.scoap3.org/record/9789/files/fulltext.xml</subfield>
</datafield>
```

„s" – size of a file
„u" – URL to the file

Example in pseudocode:

```
Foreach datafield[tag=„856”] as file
{
      download_to(file.subfield(„u”), ~/mydownloads/scoap3);
}
```

http://repo.scoap3.org/record/9789/export/xm

What to do with metadata?

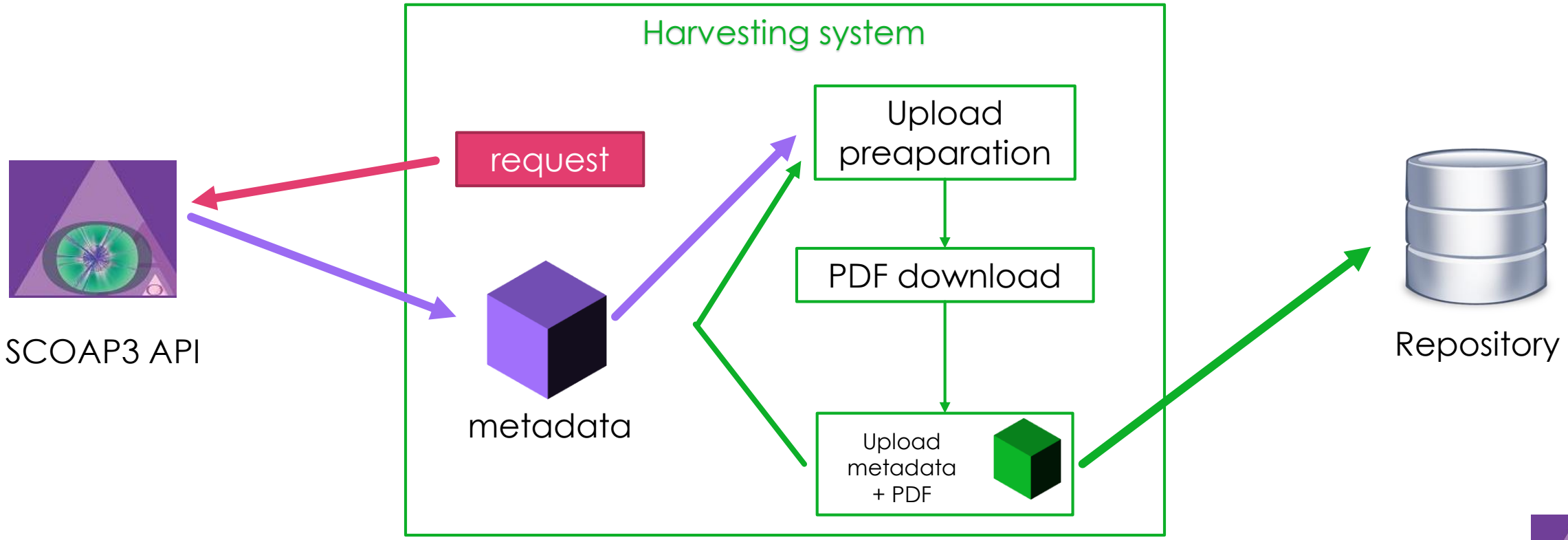
METADATA SCHEMA

USE CASES

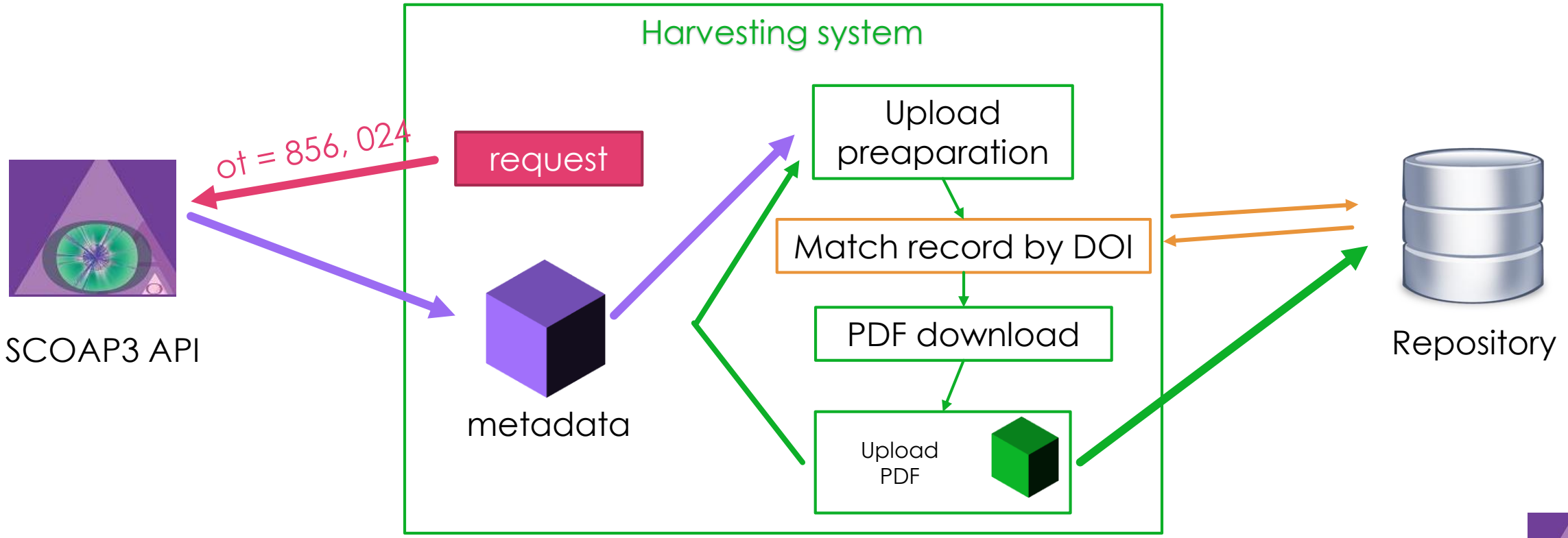
Metadata schema

- Documentation: http://scoap3.org/scoap3-repository/oai-pmh-feed
- The most important tags:
  - 024 – DOI
  - 037 – arXiv
  - 100 – first author
  - 700 – other authors
  - 245 – title
  - 773 – journal information (journal name, volum, issue, pages, year)

Theoretical example – harvest everything



Theoretical example – harvest PDFs



SCOAP3 Examples

For implementation examples from SCOAP3 partner you can visit:

github.com/SCOAP3/examples

Geneva University

Example by Jean-Blaise Claivaz

Background

- Analysis done in summer 2014 on data from
  - SCOAP3-like data
  - Authors from swiss institutions
  - Published in 2011 -> 2013
- Calculate the relative size of each institution
- Share the swiss financial participation among the partners in relation with the number of publications and not subscriptions

Future development

- Complement the analysis with data from 2014
- Use SCOAP3 repository instead of InSPIRE HEP
- Use of the API

BUT

- Work done only once a year

INFN Open Access Repository

EXAMPLE BY

ROBERTO BARBERA, INFN

About Open Access Repository

- The Open Access Repository (OAR) is a pilot data preservation repository of INFN and other Italian Research Organisations' products (publications, software, data, etc.) meant to serve both researchers and citizen scientists and to be interoperable with other related initiatives both in Italy and abroad

- The possibility to ingest INFN-(co-)authored papers/data from other highly reputed repositories is considered a very important functionality of INFN OAR

- Thanks to the recently released RESTful API's, ingestion of resources from SCOAP^3 is so straightforward that it has been automatized

Automatic ingestion of SCOAP³ data (1/3)

- Step 1: Call SCOAP³ API

```
public static HttpMethod callAPISCOAP3(String date,int jrec, int num_rec) {
    HttpMethod method =null;
    method=new GetMethod("http://repo.scoap3.org/search?of=xm&p=datecreated:"+date+"&jrec=" + jrec + "&rg=" + num_rec);
    return method;

}
```

  - Connects to the SCOAP³ HTTP endpoint
  - Retrieves in `of=xm` format `num_rec` records, starting from `jrec`, that have been created after `date`
    - e. g.: `date= 2014-11-05:00:00:00->9999-01-01`
      - retrieves records registered in SCOAP³ between 12.00 AM on the 5th of November 2014 and the moment the API is called

Automatic ingestion of SCOAP^3 data (2/3)

- Step 2: Get SCOAP^3 records

```
public static void getRecordsScoap3(String date,int jrec, int num_rec) {
    String responseXML = null;
    HttpClient client = new HttpClient();
    HttpMethod method = callAPISCOAP3(date,jrec, num_rec);

    try {
        client.executeMethod(method);

        if (method.getStatusCode() == HttpStatus.SC_OK) {

            method.getResponseBody();
            responseXML = convertStreamToString(method.getResponseBodyAsStream());
            FileWriter fw = new FileWriter("MARCXML_SCOAP3_"+date+"/marcXML_scoap3_" + jrec + "_" + num_rec + ".xml");

            fw.append(responseXML);

            fw.close();


        }
    } catch (IOException e) {
        e.printStackTrace();

    } finally {

        method.releaseConnection();

    }


}
```

  - Saves retrieved data in a local XML file

Automatic ingestion of SCOAP^3 data (3/3)

- Step 3: Format SCOAP^3 records to comply with Open Access Repository schema
  - Strips SCOAP^3 index number out
  - Modifies several MarcXML tags
  - Adds other MarcXML tags specific for Open Access Repository
  - Saves results in another local XML file
- Step 4: Ingest reformatted data in Open Access Repository

Summary

- INFRASTRUCTURE
- CONNECTING
- URL-ENCODING
- VIRTUAL FIELDS
- PARAMETERS
- METADATA SCHEMA
- LINKS

Summary

- API details: registration, infrastructure, suport, training
- Creating a query: search pattern, parameters, virtual fields, signature
- URL-encoding of parameters values
- Simple techniques of using API: Wget, cURL
- Important links:
  - API documentation: http://scoap3.org/scoap3-repository/xml-api
  - MARCXML tags: scoap3.org/scoap3-repository/oai-pmh-feed024
  - SCOAP3 examples: https://github.com/SCOAP3/examples

Thank you

Do you have questions?