What are Scopus APIs and how are these used?

Introduction to Scopus APIs

Kai Wan
Senior Product Manager, Scopus
Why are Scopus APIs Important?

- As an abstracting and indexing database, **Scopus captures articles** being published in **virtually all scholarly journals of any significance in the world**; and its profiling of authors and institutions makes it easy to find new articles by those authors at those institutions.

- The **Scopus UI (user interface)** offers many features to that end, allowing **librarians, researchers, developers and & business intelligence groups** to manually find publications originating from their institution that they can then add to their systems.

- Aside from that user interface, Scopus also **has Application Programming Interfaces (APIs)** that offer the same features, but then in a **machine-readable format that enables software, rather than humans on the UI (User Interface), to find articles, authors and institutions in Scopus**.

- This allows developers to write programs that **automatically extract data** from Scopus, and add that data to their systems.
Before we talk about APIs, let’s go back to the Scopus Data Model

The Scopus data model is designed around the notion that *articles* are written by *authors* that are *affiliated with institutions*. Visually and rather simplistically, this relational model can be represented like this:

![Simplified Scopus Data Model](image)

*Figure 1 – simplified Scopus data model*
Example of an API Request

• I am looking to retrieve records from Scopus authored by anyone with the last name Brown written in Chemistry journals.

• I sign up for my developer key at developers.elsevier.com
• I register a new project and read the documentation to learn how to structure the request.

• http://api.elsevier.com/content/search/index:SCOPUS?query=AUTHLASTNAME%28bro
  wn%29%20AND%20SUBJAREA%28CHEM%29&apikey=feada8950b5eb3c481f48762bde
  e05c6
What does the output of an API Call look like?
## Brief overview of Scopus API settings

<table>
<thead>
<tr>
<th>#</th>
<th>API Name</th>
<th>Enabled or Disabled</th>
<th>Non-subscriber</th>
<th>Subscriber</th>
<th>Weekly request quota</th>
<th>Requests/second</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Serial Title</td>
<td>Enabled</td>
<td>STANDARD, COVERIMAGE views / Default 25 results / Max 200 results</td>
<td>STANDARD, COVERIMAGE, ENHANCED Default 25 results / Max 200 results</td>
<td>20,000</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>Citations Count Metadata</td>
<td>Disabled for non-subscribers</td>
<td>N/A</td>
<td>STANDARD view / Default 25 results / Max 200 results</td>
<td>50,000</td>
<td>18</td>
</tr>
<tr>
<td>3</td>
<td>Citations Overview</td>
<td>Disabled</td>
<td>N/A</td>
<td>STANDARD view / Default 25 results / Max 200 results</td>
<td>20,000</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>Subject Classifications</td>
<td>Enabled</td>
<td>No restrictions</td>
<td>No restrictions</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>5</td>
<td>Abstract Retrieval</td>
<td>Enabled</td>
<td>META view</td>
<td>All views, default FULL view</td>
<td>10,000</td>
<td>6</td>
</tr>
<tr>
<td>6</td>
<td>Affiliation Retrieval</td>
<td>Enabled</td>
<td>N/A</td>
<td>All views, default STANDARD view</td>
<td>5,000</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td>Author Retrieval</td>
<td>Enabled</td>
<td>N/A</td>
<td>All views, default STANDARD view</td>
<td>5,000</td>
<td>3</td>
</tr>
<tr>
<td>8</td>
<td>Affiliation Search</td>
<td>Enabled</td>
<td>N/A</td>
<td>Default 25 results / Max 200 results</td>
<td>5,000</td>
<td>3</td>
</tr>
<tr>
<td>9</td>
<td>Author Search</td>
<td>Enabled</td>
<td>N/A</td>
<td>Default 25 results / Max 200 results</td>
<td>5,000</td>
<td>3</td>
</tr>
<tr>
<td>10</td>
<td>Scopus Search</td>
<td>Enabled</td>
<td>STANDARD view / Default 25 results</td>
<td>STANDARD view / Max 200 results COMPLETE view / Max 25 results</td>
<td>20,000</td>
<td>6</td>
</tr>
</tbody>
</table>
# What are the most common uses of Scopus APIs?

<table>
<thead>
<tr>
<th>Main use case</th>
<th>Scopus API use cases</th>
<th>API</th>
</tr>
</thead>
<tbody>
<tr>
<td>Showcase your achievements on your website</td>
<td>Showing publications from Scopus on your website</td>
<td>Scopus Search API</td>
</tr>
<tr>
<td></td>
<td>Showing Scopus cited-by counts on your website</td>
<td>Scopus Search API / Abstract Citation Count API</td>
</tr>
<tr>
<td>Populate and integrate with your own institutional repository</td>
<td>Populating IRs with basic document metadata from Scopus</td>
<td>Scopus Search API / Abstract Retrieval / Citation Count API</td>
</tr>
<tr>
<td></td>
<td>Populating publication histories of VIVO profiles</td>
<td>Scopus Search API / Abstract Retrieval / Citation Count API</td>
</tr>
<tr>
<td>Integrate with your library pages</td>
<td>Showing CiteScore, SJR and SNIP on journal homepage</td>
<td>Serial Title API</td>
</tr>
<tr>
<td></td>
<td>Federated search</td>
<td>Scopus Query API</td>
</tr>
</tbody>
</table>
Examples use cases for API integration

**Research and development strategy:**
I need to a better research direction for my department, which research topics are trending at the moment and will gain traction with the wider research community?

**Investment and funding:**
I need to justify the funding I received in the last few years by showing the impact of the funded research work. This impacts my future grant applications.

**Researcher assessment:**
I need to identify which researchers are the most impactful within my institution. I also need to attract more top talents for my departments.
Focus: clearer positioning of the Scopus API
Scopus API objectives

1. Add value to Scopus subscription (specifically for existing and potential Scopus API users) to increase “stickiness”
   • Align Research Products metrics for research outputs in Scopus UI with API.
   • Increase awareness of SC API for Scopus customers.
   • Provide support for increasing API usage for active Scopus API accounts

2. Improve support for API users/developers
   • Provide an API tech manual (in multiple languages) for easy (CRIS) integration
   • Provide template scripts for using API search and retrieval
   • Transfer API tech support ownership to Scopus tech team.
Scopus API objectives

3. Improve customer awareness
   • Work with marketing to establish a more simplified, logical market message for Scopus APIs in relation to assessment needs: more focus on the non-technical customer, e.g. research office.
   • Increase awareness of Scopus API integration benefits to customers.

4. Scopus team has full ownership and control over API
   • New API framework is in place in Scopus backend.
   • Author and Affiliation search API is migrated to new framework as MVP.
Thank you!
Appendix
Pybliometrics by Max Planck Institute

Pybliometrics is a python-based API wrapper for the Scopus RESTful API written for Python 3. The wrapper allows users to access the Scopus database via user-friendly interfaces and can be used without prior knowledge of RESTful APIs. The package provides classes to interact with different Scopus APIs to retrieve information as diverse as citation counts, author information or document abstracts. Files are cached to speed up subsequent analysis. The package addresses all users of Scopus data, such as researchers working in Science of Science or evaluators. It facilitates re-producibility of research projects and enhances data integrity for researchers using Scopus data.


Full article is available via SSRN - https://ssrn.com/abstract=3320470
Examples pybliometrics implementation

Automated download of information - The following code provides the time necessary to download all publications in the journal Science in 2010.

```python
from datetime import datetime
import pandas as pd
from pybliometrics.scopus import ScopusSearch

# Download
start = datetime.now().replace(microsecond=0)
s = ScopusSearch('ISSN(0036-8075) AND PUBYEAR IS 2010')
end = datetime.now().replace(microsecond=0)
print(end-start)

# 0:04:29
print(len(s.results))  # Number of papers
# 2260

# Reusing
start = datetime.now().replace(microsecond=0)
s = ScopusSearch('ISSN(0036-8075) AND PUBYEAR IS 2010')
end = datetime.now().replace(microsecond=0)
print(end-start)

# 0:00:00
```

The total number of results (i.e. the number of publications) equals 2260, which is already more than a manual download allows. Depending on download speed, the time to download the information takes less than 5 minutes. However, since pybliometrics caches the information, reusing it takes less than second only.

Examples pybliometrics implementation

Creating a collaboration network - The following example uses NetworkX (Hagberg et al., 2004) (Version 2.3) to create the co-author network for the SoftwareX journal:

```python
from itertools import combinations
import networkx as nx # Version 2.3
import matplotlib.pyplot as plt # Version 2.0.3
from pybliometrics.scopus import ScopusSearch

# Obtain authors
authors = [author_ids.split('!') for i in res]
# Create list of pairwise combinations
combs = list(combinations(a, 2) for i in authors)
# Create network
G = nx.Graph()
G.add_edges_from(combs)
print(nx.info(G))
# Type: Graph
# Number of nodes: 734
# Number of edges: 1636
# Average degree: 4.45
G = draw and save network
G.draw(
    node_size=3)
plt.savefig('network.pdf', bbox_inches='tight', figsize=(50, 50))
```

The resulting graph is depicted in figure 1. The network consists of 734 unique authors that appear on 228 papers. The nodes in the graph are connected through 1636 edges. Because we use author profile IDs provided by Scopus, we do not need to disambiguate author names.

Figure 1: Collaboration network for the SoftwareX journal