Knowledge Sharing sessions: a hands-on, crash course on Scopus APIs

Ju Chen
Consultant, Research Intelligence

Feb 2018
Agenda

- What is an API
- The Scopus APIs
- Understanding and using the Scopus APIs
- A few useful tips and tricks
- Other APIs of interest
What is an API?

Application Programming Interface

It is a way to enable software programs, rather than humans, to query Scopus.

Scopus

Javascript APIs

Federated Search APIs

Scopus Restful APIs
What is an API

Typical Keyboard Search

API Search

VS

Scopus

Scopus
Scopus APIs: how to apply

Go to dev.elsevier.com

Click on My API key

Log in with your Scopus account

Register a new site (use www.elsevier.com)

Get the API key

Let’s do it…
Services available on the Scopus APIs

Authors

- Search
- Retrieval
- Feedback*

Publications

- EID

Sources

- SOURCE-ID

Affiliations

- AFFILIATION-ID

Actions

- Search
- Retrieval

Affiliations

- ACTIONS

Actions

- Metadata

(*) Not enabled by default
RESTful interface

- Every service is a web resource identified by a unique URL
- URLs are built as follows:
  - Fixed part: http://api.elsevier.com/content
  - Middle part depending on action:
    - Search
    - Retrieve
    - Metadata
  - Final part depending on what is being searched / retrieved
  - Query parameters (api key, search query, paging, sorting, fields to be returned,…)
- Check documentation at http://api.elsevier.com for all details
- Resources can be queried using a web browser

http://api.elsevier.com/content/{action}/{object}?apiKey=your_api_key&param1=value1&param2=value2&…

REST stands for REpresentational State Transfer, it basically means that each resource is identified by a URL, it can be queried with a browser, it is stateless (i.e. the server forgets what it’s been asked for previously)
Query parameters

- Query parameters are separated from the base URL by a `?`
- Each parameter has a name and a value
- Format is `name=value`
- Parameters are separated from each other using a `&`, order does not matter
- Mandatory parameter (in order to get anything meaningful): `apiKey`
- Search parameters:
  - `query`: basically anything you can run in advanced search
  - `count`: number of results per page. Max: 200 (100 for complete view)
  - `start`: index of the first record to visualize (like paging in scopus.com)
  - `sort`: sorting criteria for results (multiple criteria comma separated)
  - `view`: select predefined view (i.e. more or less metadata)
  - `field`: return only selected fields (comma separated)
- Retrieval parameters:
  - `view`
  - `field`
Examples: publications search

• Search publications of type article, review or conference paper from France, from 2010 to 2014 included, only EID, title and citations, sort by citations descending, give me the first 200 results

http://api.elsevier.com/content/search/scopus?apiKey=0c5a3ec7fc146bd542915255233db006&query=affilcountry(france) and pubyear aft 2009 and pubyear bef 2015 and (doctype(ar) or doctype(re) or doctype(cp))&field=eid,title,citedby-count&sort=-citedby-count&count=200

• Now give me the second page of results

http://api.elsevier.com/content/search/scopus?apiKey=0c5a3ec7fc146bd542915255233db006&query=affilcountry(france) and pubyear aft 2009 and pubyear bef 2015 and (doctype(ar) or doctype(re) or doctype(cp))&field=eid,title,citedby-count&sort=-citedby-count&count=200&start=200
More examples: retrieval

• Publication:

http://api.elsevier.com/content/abstract/eid:2-s2.0-74249095519?apiKey=0c5a3ec7fc146bd542915255233db006

• Affiliation:

http://api.elsevier.com/content/affiliation/affiliation_id:60074688?apiKey=0c5a3ec7fc146bd542915255233db006

• Author:

http://api.elsevier.com/content/author/author_id:16175002400?apiKey=0c5a3ec7fc146bd542915255233db006

• Author, metrics view:

http://api.elsevier.com/content/author/author_id:16175002400?apiKey=0c5a3ec7fc146bd542915255233db006&view=metrics
How to handle the data

XML data usually needs to be mapped to tables.

Key issue, normally accomplished by coding.

Excel can be used here, to some extent.

You can save the results of a search query as an XML file and open it with Excel (as an XML table or read-only workbook).

Let’s try this…
Some useful tools to work with the APIs

**REST Client**: Firefox add on to work with REST services: https://addons.mozilla.org/en-GB/firefox/addon/restclient/

**WGet** (portable version): download search results without using the browser: http://portableapps.com/apps/internet/winwget_portable

**SOAPUI**: to work with SOAP web services (like the matching services): http://www.soapui.org/

**Notepad++**: Text editor with XML formatting / validating capabilities and a powerful search / replace feature: https://notepad-plus-plus.org/
More APIs: Matching Services (SciVal Analytics)

- SOAP API (not REST)
- Used in the REF, ERA and in Italy
- Performs multiple searches in one go to match a publication metadata with the corresponding record in Scopus.com
- Dedicated API key
More APIs: Percentiles API (SciVal Analytics)

• REST API
• Provides “continuous” citation and journal metric percentiles normalized by year, subject area and document type
• Requires dedicated API key (same as matching services)
• Snapshots taken every quarter
• Previous snapshots can be queried
• Snapshot coded as <year>Q<quarter> e.g.: 2016Q1

http://sais.scivalcontent.com/REST/percentiles/2015Q4/2-s2.0-35748934247/?clientKey=4c7fc887a8163a753f167e082c66da3f
Percentiles API: example response

```xml
<?xml version="1.0"?>
<percentileResponse snapshot="2015Q4">
  <document eid="2-s2.0-35748934247" docType="ar" year="2011" sourceId="12100156717">
    <citationCount>17</citationCount>
    <ASJC code="2404"/>
    <ASJC code="2405"/>
    <ASJC code="2406"/>
  </document>
  <CitationPercentiles docType="ar" year="2011" bestCitationPercentileMatched="17.792">
    <ASJC code="2404" bestPercentileMatched="19.266"/>
    <ASJC code="2405" bestPercentileMatched="17.792"/>
    <ASJC code="2406" bestPercentileMatched="25.504"/>
  </CitationPercentiles>
  <JournalPercentiles sourceId="12100156717" year="2011">
    <JournalPercentile metric="SNIP" value="0.276" bestPercentileNonWeight="82.759" bestPercentileDocWeight="91.629">
      <ASJC code="2404" percentileNonWeight="83.193" percentileDocWeight="91.629"/>
      <ASJC code="2405" percentileNonWeight="85.714" percentileDocWeight="95.004"/>
      <ASJC code="2406" percentileNonWeight="82.759" percentileDocWeight="95.162"/>
    </JournalPercentile>
    <JournalPercentile metric="SJR" value="0.310" bestPercentileNonWeight="74.790" bestPercentileDocWeight="84.780">
      <ASJC code="2404" percentileNonWeight="74.790" percentileDocWeight="84.780"/>
      <ASJC code="2405" percentileNonWeight="75.000" percentileDocWeight="90.008"/>
      <ASJC code="2406" percentileNonWeight="77.966" percentileDocWeight="93.967"/>
    </JournalPercentile>
    <JournalPercentile metric="IPP" value="0.649" bestPercentileNonWeight="75.862" bestPercentileDocWeight="86.470">
      <ASJC code="2404" percentileNonWeight="78.992" percentileDocWeight="86.470"/>
      <ASJC code="2405" percentileNonWeight="76.786" percentileDocWeight="90.841"/>
      <ASJC code="2406" percentileNonWeight="75.862" percentileDocWeight="93.706"/>
    </JournalPercentile>
  </JournalPercentiles>
</percentileResponse>
```
More APIs: Fingerprint Engine (SciVal Analytics)

- RESTful API
- Feature rich, but basic services are
  - Categorize: rank best thesauri based on the text
  - Generate fingerprint using selected thesaurus
  - It uses HTTP basic authentication (username / password)
  - Text is Posted to the service via HTTP post (just like when you submit a form to a website)

https://fingerprintengine.scivalcontent.com/Taco7600/TacoService.svc/MeSHXmlConceptsOnly
Example of Fingerprint APIs input...

Method: POST
URL: https://fingerprintengine.scivalcontent.com/Taco7600/TacoService.svc/MeSHXmlConceptsOnly

Headers:
- Header Name: Authorization
- Header Value: Basic UINTX1Jlc2VhcmNoTWdtdDo5bVRaRjIUQ==

Body:

```xml
<Text>
<Title>Mechanisms of disease: Inflammation, atherosclerosis, and coronary artery disease</Title>
<Abstract>In this review article, Göran Hansson, a pioneer in the study of the role of inflammation in atherosclerosis and coronary artery disease, summarizes new ideas on the pathogenesis of acute coronary syndromes.</Abstract>
</Text>
```
...and output

```xml
  <Annotations>
    <Annotation i:type="ConceptAnnotation">
      <AFreq>1</AFreq>
      <ConceptID>6564</ConceptID>
      <Name>Disease</Name>
      <Rank>1</Rank>
      <Thesaurus>MeSH</Thesaurus>
    </Annotation>
    <Annotation i:type="ConceptAnnotation">
      <AFreq>2</AFreq>
      <ConceptID>11307</ConceptID>
      <Name>Inflammation</Name>
      <Rank>1</Rank>
      <Thesaurus>MeSH</Thesaurus>
    </Annotation>
    <Annotation i:type="ConceptAnnotation">
      <AFreq>2</AFreq>
      <ConceptID>1903</ConceptID>
      <Name>Atherosclerosis</Name>
      <Rank>1</Rank>
      <Thesaurus>MeSH</Thesaurus>
    </Annotation>
    <Annotation i:type="ConceptAnnotation">
      <AFreq>2</AFreq>
      <ConceptID>502868</ConceptID>
      <!-- More annotations here -->
  </Annotations>
</TextAnalysis>
```
Other free APIs of interest from 3rd parties: ORCID

• ORCID profile
  - http://pub.orcid.org/0000-0002-2016-1966/orcid-profile

• ORCID publications
  - http://pub.orcid.org/0000-0002-2016-1966/orcid-works

• Visit http://members.orcid.org/api/introduction-orcid-member-api for more info on the ORCID APIs.
Other free APIs of interest from 3rd parties: PubMed

- REST API (kind of…)
- Uses the PubMed Id to identify the papers
- Example of a search:


- Example of a retrieval

Other free APIs of interest from 3rd parties: Mendeley

• A bit more complicated due to Authentication and Authorization protocol (OAUTH2) but it’s available and full of interesting stuff.
• Info at http://dev.mendeley.com/
Putting it all together: My Mendeley fingerprint map

Mendeley APIs (bibliography)

Gephi (co-occurrence network, clustering)

D3.js (visualization)

My semantic map (click me)

Scopus APIs (abstracts)

Fingerprint Engine APIs (fingerprints)
Thanks! Questions?