arXiv / Help / arXiv API

arXiv API

This is the home site of the arXiv API. The goal of the API is to allow application developers access to all of the arXiv data, search and linking facilities with an easy-to-use programmatic interface. This page provides links to developer documentation, and gives instructions for how to join the mailing list and contact other developers and maintainers.

Please review the Terms of Use for arXiv APIs before using the arXiv API.

For more news and questions about the arXiv API, please see our arxiv-api group. Additional information is also in the API FAQ.

About the arXiv API

The Cornell University e-print arXiv, hosted at arXiv.org, is a document submission and retrieval system that is heavily used by the physics, mathematics and computer science communities. It has become the primary means of communicating cutting-edge manuscripts on current and ongoing research. The open-access arXiv e-print repository is available worldwide, and presents no entry barriers to readers, thus facilitating scholarly communication. Manuscripts are often submitted to the arXiv before they are published by more traditional means. In some cases they may never be submitted or published elsewhere, and in others, arXiv-hosted manuscripts are used as the submission channel to traditional publishers such as the American Physical Society, and newer forms of publication such as the Journal for High Energy Physics and overlay journals.

The primary interface to the arXiv has been human-oriented html web pages. The purpose of the arXiv API is to allow programmatic access to the arXiv's e-print content and metadata. The goal of the interface is to facilitate new and creative use of the the vast body of material on the arXiv by providing a low barrier to entry for application developers.

Quickstart

API calls are made via an HTTP GET or POST requests to an appropriate url. For example, the url

http://export.arxiv.org/api/query?search_query=all:electron

retrieves results that match the search query `all:electron`. This url can be accessed from any web-enabled client including your web browser, or via web libraries common to almost all programming languages. There is no arXiv-supplied software that must be downloaded and installed to be able to use the api.

Please see Using the arXiv API, or the User's Manual for more information.

Using the arXiv API

Since the arXiv API is based on the now ubiquitous HTTP, using it should be fairly straight forward from the programming language of your choice. The primary access point for the api is a url that encodes your desired search parameters. For example, the url:

http://export.arxiv.org/api/query?search_query=all:electron&start=0&max_results=10

indicates that you want to use the api query interface to retrieve the first ten results that match the query `all:electron`. This url calls the api, which returns the results in the Atom 1.0 format.

Atom 1.0 is an xml-based format that is commonly used in website syndication feeds. It is lightweight, and human readable, and results can be cleanly read in many web browsers. For detailed information on Atom, you can read the official Atom 1.0 specification.

We recommend that you use a web browser such as to play around with constructing the api url's to get a feel for how the system works. This is also a great debugging tool to make sure your url's make sense to the api. Firefox renders Atom particularly cleanly. More detailed documentation on constructing API urls can be found in the User's Manual.

Once you have familiarized yourself with the api, you should be able to easily write programs that call the API automatically. Most programming languages, if not all, have libraries that allow you to make HTTP requests. Since Atom is growing, not all languages have libraries that support Atom parsing, so most of the programming effort will be in digesting the responses you receive. The languages that we know of that can easily handle calling the api via HTTP and parsing the results include:

- Perl (via XML::Atom) (example)
- Python (via feedparser) (example)
- Ruby (via feedtools) (example)
- PHP (via SimplePie) (example)

Below we include code snippets for these languages that perform the bare minimum functionality - calling the api and printing the raw Atom results. See the documentation and example programs for more detailed examples. If your favorite language is not up here, write us with an example, and we'll be glad to post it! For more detailed examples in these languages which cover more advanced API programming, please see the User's Manual.

All of the simple examples produce an output which looks like:

Example: A Typical Atom Response (wrapped for ease of reading, issue query now)

```
<?xml version="1.0" encoding="utf-8"?>
<feed xmlns="http://www.w3.org/2005/Atom"
      xmlns:opensearch="http://a9.com/-/spec/opensearch/1.1/"
      xmlns:arxiv="http://arxiv.org/schemas/atom">
  <link xmlns="http://www.w3.org/2005/Atom"
   href="http://export.arxiv.org/api/query?search_query=all:electron&amp;id_list=&amp;start=0&amp;max_results=1"
   rel="self" type="application/atom+xml"/>
  <title xmlns="http://www.w3.org/2005/Atom">ArXiv Query:
search_query=all:electron&amp;id_list=&amp;start=0&amp;max_results=1</title>
  <id xmlns="http://www.w3.org/2005/Atom">http://arxiv.org/api/cHxbiOdZaP56ODnBPIenZhzg5f8</id>
  <updated xmlns="http://www.w3.org/2005/Atom">2007-10-08T00:00:00-04:00</updated>
  <opensearch:totalResults xmlns:opensearch="http://a9.com/-/spec/opensearch/1.1/">1000</opensearch:totalResults>
  <opensearch:startIndex xmlns:opensearch="http://a9.com/-/spec/opensearch/1.1/">0</opensearch:startIndex>
  <opensearch:itemsPerPage xmlns:opensearch="http://a9.com/-/spec/opensearch/1.1/">1</opensearch:itemsPerPage>
  <entry xmlns="http://www.w3.org/2005/Atom" xmlns:arxiv="http://arxiv.org/schemas/atom">
    <id xmlns="http://www.w3.org/2005/Atom">http://arxiv.org/abs/hep-ex/0307015</id>
    <published xmlns="http://www.w3.org/2005/Atom">2003-07-07T13:46:39-04:00</published>
    <updated xmlns="http://www.w3.org/2005/Atom">2003-07-07T13:46:39-04:00</updated>
    <title xmlns="http://www.w3.org/2005/Atom">Multi-Electron Production at High Transverse
Momenta in ep Collisions at HERA</title>
    <summary xmlns="http://www.w3.org/2005/Atom">
  Multi-electron production is studied at high electron transverse momentum in
positron- and electron-proton collisions using the H1 detector at HERA. The
data correspond to an integrated luminosity of 115 pb-1. Di-electron and
tri-electron event yields are measured. Cross sections are derived in a
restricted phase space region dominated by photon-photon collisions. In general
good agreement is found with the Standard Model predictions. However, for
electron pair invariant masses above 100 GeV, three di-electron events and
three tri-electron events are observed, compared to Standard Model expectations
of 0.30 \pm 0.04 and 0.23 \pm 0.04, respectively.
</summary>
    <author xmlns="http://www.w3.org/2005/Atom">
      <name xmlns="http://www.w3.org/2005/Atom">H1 Collaboration</name>
    </author>
    <arxiv:comment xmlns:arxiv="http://arxiv.org/schemas/atom">23 pages, 8 figures and 4 tables</arxiv:comment>
    <arxiv:journal_ref xmlns:arxiv="http://arxiv.org/schemas/atom">Eur.Phys.J. C31 (2003) 17-29</arxiv:journal_ref>
    <link xmlns="http://www.w3.org/2005/Atom"
```

```
      href="http://arxiv.org/abs/hep-ex/0307015v1" rel="alternate" type="text/html"/>
     <link xmlns="http://www.w3.org/2005/Atom" title="pdf"
      href="http://arxiv.org/pdf/hep-ex/0307015v1" rel="related" type="application/pdf"/>
     <arxiv:primary_category xmlns:arxiv="http://arxiv.org/schemas/atom"
      term="hep-ex" scheme="http://arxiv.org/schemas/atom"/>
     <category term="hep-ex" scheme="http://arxiv.org/schemas/atom"/>
   </entry>
 </feed>
```

Perl

LWP is in the default perl installation on most platforms. It can be downloaded and installed from CPAN. Sample code to produce the above output is:

```
use LWP;
use strict;

my $url = 'http://export.arxiv.org/api/query?search_query=all:electron&start=0&max_results=1';
my $browser = LWP::UserAgent->new();
my $response = $browser->get($url);
print $response->content();
```

Python

The urllib module is part of the python standard library, and is included in any default installation of python. Sample code to produce the above output is:

```
import urllib
url = 'http://export.arxiv.org/api/query?search_query=all:electron&start=0&max_results=1'
data = urllib.urlopen(url).read()
print data
```

Ruby

The net/http and uri modules are part of the ruby standard library, and are included in any default installation of ruby. Sample code to produce the above output is:

```
require 'net/http'
require 'uri'
url = URI.parse('http://export.arxiv.org/api/query?search_query=all:electron&start=0&max_results=1')
res = Net::HTTP.get_response(url)
print res.body
```

PHP

The file_get_contents() function is part of the PHP core language:

```
<?php
$url = 'http://export.arxiv.org/api/query?search_query=all:electron&start=0&max_results=1';
$response = file_get_contents($url);
print_r($response);
?>
```

arXiv API documentation

The API User's Manual discusses the API interface and returned Atom format in detail, and gives code examples in the languages listed above. The API FAQ has answers to common questions, or feel free to ask the arxiv-api discussion list.

Community

We would love to know how you are using the arXiv API. Please send us an email to the mailing list to tell us about your project, and what language/library you are using. Please include a url of your project, and we will post a link to it from this page.

The best way to learn about the arXiv API, and to get help from others is to join our mailing list. It is better to ask your questions in this forum for all to see. More often than not, someone else has the same question, or can provide an answer. Questions and feedback are the primary information channel that we use to improve this service.

Projects Using the API

The following projects use the arXiv API:

- OpenWetWare's Mediawiki Installation
- Sonny Software's Bookends Reference Manager for OSX
- arXiv Droid - arXiv app for Android
- Retrieve Bibliographic arXiv Information
- The snarXiv
- daily arXiv by categories
- PaperRater.org - a web-based tool for open review and social reading of scientific publications
- ArXiv Analytics - a web portal for reading and discussing arXiv eprints
- Bibcure - keeps bibtex up to date and normalized, and allows you to download all papers inside your bibtex
- biblio.el - Download BibTeX entries from arXiv and others in Emacs
- Lib arXiv - arXiv app for iOS devices
- arxivist.com

Helping Out

We love to hear from you. If you have changes to the documentation, code examples, example applications that use the API, and general comments, please send them to the mailing list. We value all of your discussion.

Mailing List

To join the mailing list, please visit our arxiv-api groups page. Anyone can join, and we encourage you to do so.

"arXiv API" revision 0.5.2. Last modified 2019-09-10.