Using the PASTA+ Search API To Create a Local Data Catalog

Duane Costa
Environmental Data Initiative

PASTA+ Search API: Motivation for Using It to Create a Local Data Catalog

- Simple inventory (no user query input)
    - Need to construct a query that acts like a “fish net”, hauling in all data packages from the deep blue sea of the PASTA+ repository, while leaving everything else behind
- Search Interface
    - Simple search interface
        - Query terms that the user inputs into a form field
    - Custom search interface
        - Users query PASTA+ in a way that differs from the search interface in the central data portal
        - Restrict search results to a particular geographic area, project title, etc.
        - User can browse on specific keywords that are meaningful to your site or project, e.g. “Pocillopora” is a meaningful search term for the Moorea Coral Reef LTER

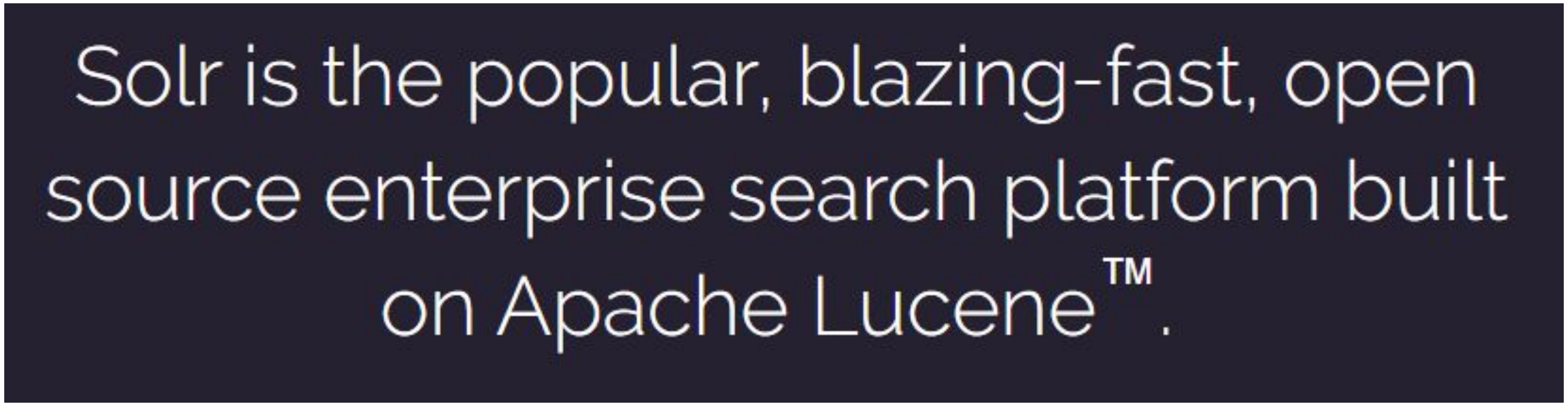
PASTA+ Search API: Advantages WHEN Using It to Create a Local Data Catalog

- Automatically synchronized with contents of the central repository, avoiding inconsistencies.
- Lower maintenance. Let PASTA+ automatically retrieve the list of data packages it holds for your local site instead of manually maintaining this list on your own
- Retain local styling and branding

Design and Implementation Decisions for a Local Data Catalog

- The user interface -- simple list or support for user query?
- Which programming language or CMS to interact with PASTA+ web services?
- How to transform the XML search results to HTML?
- How to construct links back to the data

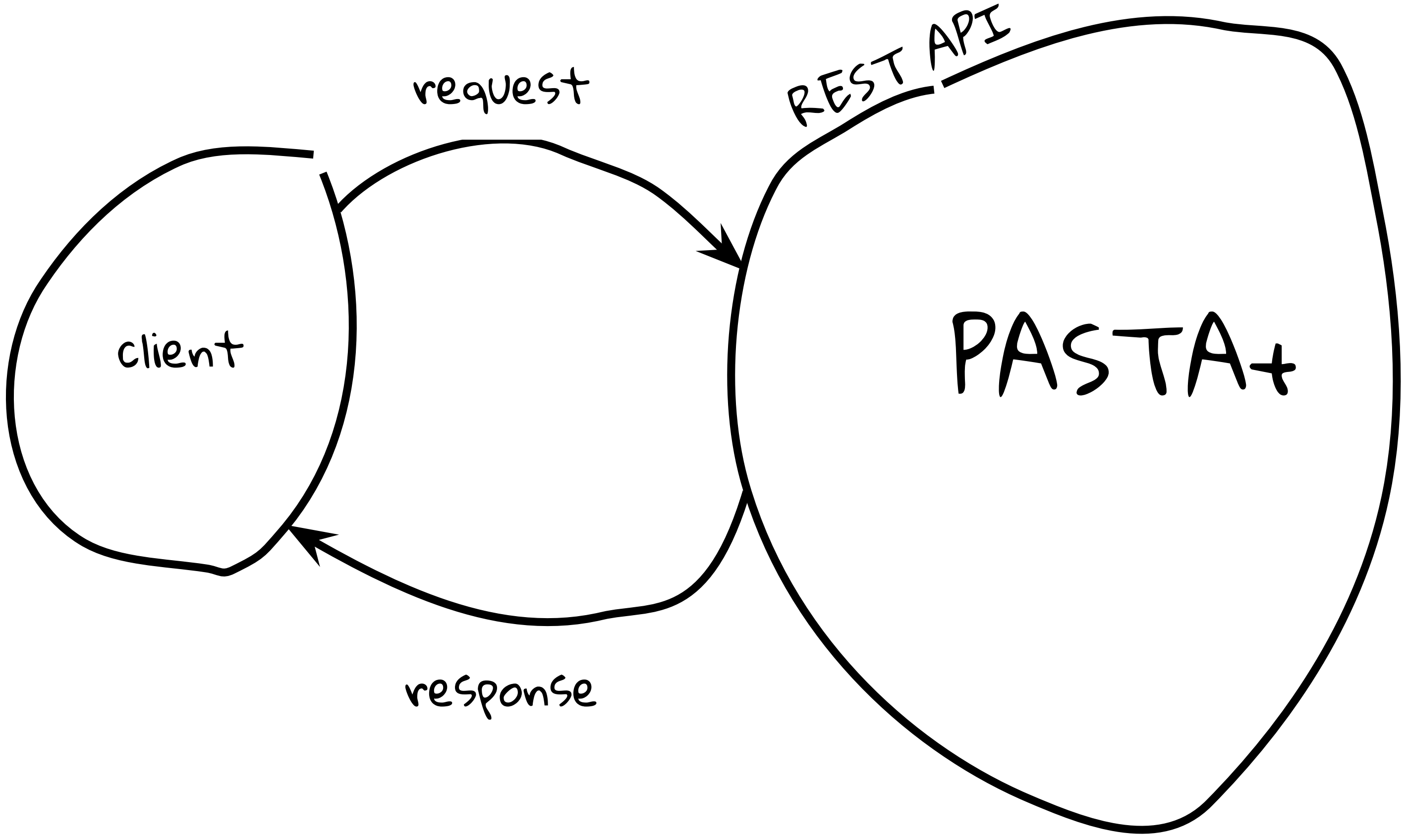
PASTA+ Search API Is Built on Solr


Solr is the popular, blazing-fast, open source enterprise search platform built on Apache Lucene™.

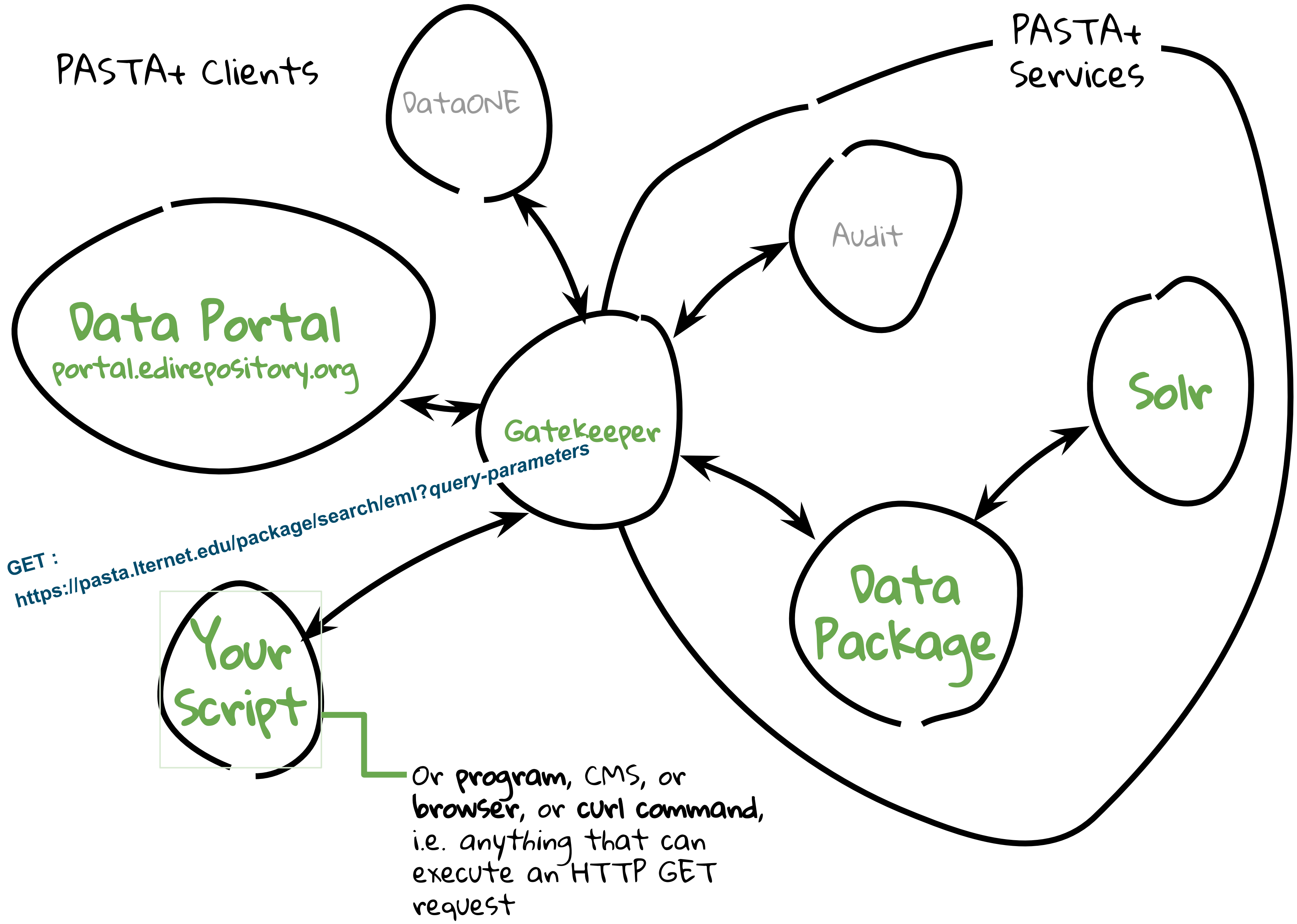
http://lucene.apache.org/solr/

PASTA+ Search API is JUST another Web Service

(though I might argue, kind of special)



A highly technical diagram


PASTA+ Clients
DataONE
PASTA+ Services
Audit
Data Portal
portal.edirepository.org
Gatekeeper
Solr
GET : https://pasta.lternet.edu/package/search/eml?query-parameters
Data Package
Your Script
Or program, CMS, or browser, or curl command, i.e. anything that can execute an HTTP GET request

PASTA+ Search API: Three Simple Steps to Understanding

1. **Indexing.** What metadata fields can I query?

   Note that not all EML metadata values are indexed. For example:

   ```
   <formatString>YYYY-MM-DD</formatString>
   ```

   We have to ask ourselves, would anyone really want to search on "YYYY-MM-DD"? If the answer is "no", then why bother indexing it?

2. **Querying.** How do I compose a query and send it to PASTA+?

3. **Search Results.** What does PASTA+ send back to me?

Indexing EML Metadata in Solr

Identifiers	Strings & Text	Temporal	Spatial
scope (e.g. edi) id (e.g. edi.1) packageid (e.g.edi.1.1) doi	title keyword author organization responsibleParties abstract methods funding taxonomic timescale geographicdescription projectTitle relatedProjectTitle subject	pubdate singledate begindate enddate	coordinates

These are all the metadata fields we currently treat as searchable content. But, of course, we could add more, if needed.

Querying Metadata: API Basics

Search Data Packages

GET : https://pasta.lternet.edu/package/search/eml?*query-parameters*

Querying Metadata: first example

```
https://pasta.lternet.edu/package/search/eml?q=keyword:disturbance&fl=id
```

- q=keyword:disturbance
  - Match only data packages that have the “disturbance” keyword
    - Case insensitive
- fl=id
  - Return only the id field for each matched document
    - e.g. “edi.1”

Only two query parameters. Easy, right?

Querying Metadata: example 2

```
curl -X GET
"https://pasta.lternet.edu/package/search/eml?q=keyword:fish&fq=title:%22North+Inlet+LTER%22&fl=doi,packageid,title,author,score&sort=score,desc&sort=packageid,asc&start=0&rows=10"
```

- The entire URL is quoted on the command line because “&” has special meaning to the shell.
- Match “fish” in the keyword field and count it toward relevance scoring
- Filter on “North Inlet LTER” (as a single search phrase) in the title field but don’t count it toward relevance scoring.
- Note the “%22” used for double-quotes and “+” used for space. Sometimes we need URL-encoding of certain characters depending on how we’re using the URL.
- Return several different fields for each matched document
  - Use fl=* to return all fields
- Sort on the relevance score, descending order
- Secondary sort on the package identifier, ascending order
- Return documents beginning at row 0
- Return up to 10 rows of matching documents (note: if omitted, defaults to 10 rows)

Querying Metadata: example 2

```
curl -X GET
"https://pasta.lternet.edu/package/search/eml?q=keyword:fish&fq=title:%22North+Inlet+LTER%22&fl=doi,packageid,title,author,score&sort=score,desc&sort=packageid,asc&start=0&rows=10"
```

- The entire URL is quoted on the command line because “&” has special meaning to the shell.
- Match “fish” in the keyword field and count it toward relevance scoring
- Filter on “North Inlet LTER” (as a single search phrase) in the title field but don’t count it toward relevance scoring.
- Note the “%22” used for double-quotes and “+” used for space. Sometimes we need URL-encoding of certain characters depending on how we’re using the URL.
- Return several different fields for each matched document
    - Use fl=* to return all fields
- Sort on the relevance score, descending order
- Secondary sort on the package identifier, ascending order
- Return documents beginning at row 0
- Return up to 10 rows of matching documents (note: if omitted, defaults to 10 rows)

Querying Metadata: example 2

```
curl -X GET
"https://pasta.lternet.edu/package/search/eml?q=keyword:fish&fq=title:%2
2North+Inlet+LTER%22&fl=doi,packageid,title,author,score&sort=score,de
sc&sort=packageid,asc&start=0&rows=10"
```

- The entire URL is quoted on the command line because "&" has special meaning to the shell.
- Match "fish" in the keyword field and count it toward relevance scoring
- Filter on "North Inlet LTER" (as a single search phrase) in the title field but don't count it toward relevance scoring.
- Note the "%22" used for double-quotes and "+" used for space. Sometimes we need URL-encoding of certain characters depending on how we're using the URL.
- Return several different fields for each matched document
    - Use fl=* to return all fields
- Sort on the relevance score, descending order
- Secondary sort on the package identifier, ascending order
- Return documents beginning at row 0
- Return up to 10 rows of matching documents (note: if omitted, defaults to 10 rows)

Querying Metadata: example 2

```
curl -X GET
"https://pasta.lternet.edu/package/search/eml?q=keyword:fish&fq=title:%2
2North+Inlet+LTER%22&fl=doi,packageid,title,author,score&sort=score,de
sc&sort=packageid,asc&start=0&rows=10"
```

- The entire URL is quoted on the command line because “&” has special meaning to the shell.
- Match “fish” in the keyword field and count it toward relevance scoring
- Filter on “North Inlet LTER” (as a single search phrase) in the title field but don’t count it toward relevance scoring.
- Note the “%22” used for double-quotes and “+” used for space. Sometimes we need URL-encoding of certain characters depending on how we’re using the URL.
- Return several different fields for each matched document
  - Use fl=* to return all fields
- Sort on the relevance score, descending order
- Secondary sort on the package identifier, ascending order
- Return documents beginning at row 0
- Return up to 10 rows of matching documents (note: if omitted, defaults to 10 rows)

Querying Metadata: example 2

```
curl -X GET
"https://pasta.lternet.edu/package/search/eml?q=keyword:fish&fq=title:%2
2North+Inlet+LTER%22&fl=doi,packageid,title,author,score&sort=score,de
sc&sort=packageid,asc&start=0&rows=10"
```

- The entire URL is quoted on the command line because “&” has special meaning to the shell.
- Match “fish” in the keyword field and count it toward relevance scoring
- Filter on “North Inlet LTER” (as a single search phrase) in the title field but don’t count it toward relevance scoring.
- Note the “%22” used for double-quotes and “+” used for space. Sometimes we need URL-encoding of certain characters depending on how we’re using the URL.
- Return several different fields for each matched document
  - Use fl=* to return all fields
- Sort on the relevance score, descending order
- Secondary sort on the package identifier, ascending order
- Return documents beginning at row 0
- Return up to 10 rows of matching documents (note: if omitted, defaults to 10 rows)

Querying Metadata: example 2

```
curl -X GET
"https://pasta.lternet.edu/package/search/eml?q=keyword:fish&fq=title:%2
2North+Inlet+LTER%22&fl=doi,packageid,title,author,score&sort=score,de
sc&sort=packageid,asc&start=0&rows=10"
```

- The entire URL is quoted on the command line because “&” has special meaning to the shell.
- Match “fish” in the keyword field and count it toward relevance scoring
- Filter on “North Inlet LTER” (as a single search phrase) in the title field but don’t count it toward relevance scoring.
- Note the “%22” used for double-quotes and “+” used for space. Sometimes we need URL-encoding of certain characters depending on how we’re using the URL.
- Return several different fields for each matched document
  - The score field is a relevance score generated by Solr, not an indexed metadata field
  - Use fl=* to return all fields
- Sort on the relevance score, descending order
- Secondary sort on the package identifier, ascending order
- Return documents beginning at row 0
- Return up to 10 rows of matching documents (note: if omitted, defaults to 10 rows)

Querying Metadata: example 2

```
curl -X GET
"https://pasta.lternet.edu/package/search/eml?q=keyword:fish&fq=title:%2
2North+Inlet+LTER%22&fl=doi,packageid,title,author,score&sort=score,de
sc&sort=packageid,asc&start=0&rows=10"
```

- The entire URL is quoted on the command line because “&” has special meaning to the shell.
- Match “fish” in the keyword field and count it toward relevance scoring
- Filter on “North Inlet LTER” (as a single search phrase) in the title field but don’t count it toward relevance scoring.
- Note the “%22” used for double-quotes and “+” used for space. Sometimes we need URL-encoding of certain characters depending on how we’re using the URL.
- Return several different fields for each matched document
  - Use fl=* to return all fields
- Sort on the relevance score, descending order
- Secondary sort on the package identifier, ascending order
- Return documents beginning at row 0
- Return up to 10 rows of matching documents (note: if omitted, defaults to 10 rows)

Querying Metadata: example 2

```
curl -X GET
"https://pasta.lternet.edu/package/search/eml?q=keyword:fish&fq=title:%2
2North+Inlet+LTER%22&fl=doi,packageid,title,author,score&sort=score,de
sc&sort=packageid,asc&start=0&rows=10"
```

- The entire URL is quoted on the command line because “&” has special meaning to the shell.
- Match “fish” in the keyword field and count it toward relevance scoring
- Filter on “North Inlet LTER” (as a single search phrase) in the title field but don’t count it toward relevance scoring.
- Note the “%22” used for double-quotes and “+” used for space. Sometimes we need URL-encoding of certain characters depending on how we’re using the URL.
- Return several different fields for each matched document
    - Use fl=* to return all fields
- Sort on the relevance score, descending order
- Secondary sort on the package identifier, ascending order
- Return documents beginning at row 0
- Return up to 10 rows of matching documents (note: if omitted, defaults to 10 rows)

Querying Metadata: example 2

```
curl -X GET
"https://pasta.lternet.edu/package/search/eml?q=keyword:fish&fq=title:%2
2North+Inlet+LTER%22&fl=doi,packageid,title,author,score&sort=score,de
sc&sort=packageid,asc&start=0&rows=10"
```

- The entire URL is quoted on the command line because "&" has special meaning to the shell.
- Match "fish" in the keyword field and count it toward relevance scoring
- Filter on "North Inlet LTER" (as a single search phrase) in the title field but don't count it toward relevance scoring.
- Note the "%22" used for double-quotes and "+" used for space. Sometimes we need URL-encoding of certain characters depending on how we're using the URL.
- Return several different fields for each matched document
  - Use fl=* to return all fields
- Sort on the relevance score, descending order
- Secondary sort on the package identifier, ascending order
- Return documents beginning at row 0
- Return up to 10 rows of matching documents (note: if omitted, defaults to 10 rows)

Querying Metadata: example 2

```
curl -X GET
"https://pasta.lternet.edu/package/search/eml?q=keyword:fish&fq=title:%2
2North+Inlet+LTER%22&fl=doi,packageid,title,author,score&sort=score,de
sc&sort=packageid,asc&start=0&rows=300"
```

- The entire URL is quoted on the command line because “&” has special meaning to the shell.
- Match “fish” in the keyword field and count it toward relevance scoring
- Filter on “North Inlet LTER” (as a single search phrase) in the title field but don’t count it toward relevance scoring.
- Note the “%22” used for double-quotes and “+” used for space. Sometimes we need URL-encoding of certain characters depending on how we’re using the URL.
- Return several different fields for each matched document
  - Use fl=* to return all fields
- Sort on the relevance score, descending order
- Secondary sort on the package identifier, ascending order
- Return documents beginning at row 0
- Return up to 300 matching documents (note: if omitted, defaults to 10 rows)

Querying Metadata: example 2

```
curl -X GET
"https://pasta.lternet.edu/package/search/eml?q=keyword:fish&fq=title:%22North+Inlet+LTER%22&fl=doi,packageid,title,author,score&sort=score,desc&sort=packageid,asc&start=0&rows=10"
```

- The entire URL is quoted on the command line because “&” has special meaning to the shell.
- Match “fish” in the keyword field and count it toward relevance scoring
- Filter on “North Inlet LTER” (as a single search phrase) in the title field but don’t count it toward relevance scoring.
- Note the “%22” used for double-quotes and “+” used for space. Sometimes we need URL-encoding of certain characters depending on how we’re using the URL.
- Return several different fields for each matched document
  - Use fl=* to return all fields
- Sort on the relevance score, descending order
- Secondary sort on the package identifier, ascending order
- Return documents beginning at row 0
- Return up to 10 rows of matching documents (note: if omitted, defaults to 10 rows)

Using Q and FQ to construct our "Fish Net"

Example fq usage	Explanation
q=*&fq=scope:knb-lter-hbr	Historically, LTER sites have a dedicated scope in the standard format "knb-lter-xyz", where "xyz" is the three-letter site abbreviation (in lowercase)
q=*&fq=scope:edi&fq=title:%22Illinois+EPA+Ambient+Lake+Monitoring+Program+(ALMP):%22	If some boilerplate text is guaranteed to always be in every title
q=*&fq=scope:edi&fq=projectTitle:%22The+University+of+Kansas+Field+Station,+Kansas,+USA%22	Perhaps the boilerplate text is more reliably found in the projectTitle element
q=*&fq=scope:edi&fq=funding:"EF-1065786"	A grant number in the funding element can be useful to match. But place it within quotes to ensure an exact match. Otherwise, you'll also get close matches which you probably don't want.

Search results XML (1 of 2)

https://pasta.lternet.edu/package/search/eml?q=packageid:knb-lter-nin.1.1&fl=*

"fl=*" means "Give me all the fields you've got stored for the matching document(s)."

```
<resultset numFound="1" start="0" rows="10">
  <document>
    <abstract>
      This data package consists of Daily Water Sample Nutrient Data for North Inlet Estuary, South Carolina, from
      1978 to 1992,
      (Truncated to save space)
    </abstract>
    <begindate>1978-09-01</begindate>
    <doi>doi:10.6073/pasta/0675d3602ff57f24838ca8d14d7f3961</doi>
    <enddate>1992-04-21</enddate>
    <funding/>
    <geographicdescription>
      North Inlet encompasses about 2,630 hectares of tidal marshes and wetlands near Georgetown, South Carolina,
      USA. (Truncated to save space)
    </geographicdescription>
    <id>knb-lter-nin.1</id>
    <methods/>
    <packageid>knb-lter-nin.1.1</packageid>
    <pubdate>2013</pubdate>
    <responsibleParties>
      North Inlet LTER NIN-LTER Vernberg, F. John Blood, Elizabeth
    </responsibleParties>
    <scope>knb-lter-nin</scope>
```

Search results XML (2 of 2)

```
    <site>nin</site>
     <taxonomic/>
    <title>
      Daily Water Sample Nutrient Data for North Inlet Estuary, South Carolina, from 1978 to 1992, North Inlet LTER
    </title>
    <authors>
      <author>Vernberg, F. John</author>
      <author>Blood, Elizabeth</author>
    </authors>
    <spatialCoverage>
      <coordinates>-79.2936 33.1925 -79.1002 33.357</coordinates>
    </spatialCoverage>
    <sources></sources>
    <keywords>
      <keyword>nutrient dynamics</keyword>
      <keyword>North Inlet Estuary</keyword>
      <keyword>Baruch Institute</keyword>
      <keyword>Georgetown, South Carolina</keyword>
    </keywords>
    <organizations>
      <organization>North Inlet LTER</organization>
      <organization>NIN-LTER</organization>
    </organizations>
    <singledates></singledates>
    <timescales></timescales>
  </document>
</resultset>
```

Search results XML (2 of 2)

```
      <site>nin</site>
      <taxonomic/>
      <title>
        Daily Water Sample Nutrient Data for North Inlet Estuary, South Carolina, from 1978 to 1992, North Inlet LTER
      </title>
      <authors>
        <author>Vernberg, F. John</author>
        <author>Blood, Elizabeth</author>
      </authors>
      <spatialCoverage>
        <coordinates>-79.2936 33.1925 -79.1002 33.357</coordinates>
      </spatialCoverage>
      <sources></sources>
      <keywords>
        <keyword>nutrient dynamics</keyword>
        <keyword>North Inlet Estuary</keyword>
        <keyword>Baruch Institute</keyword>
        <keyword>Georgetown, South Carolina</keyword>
      </keywords>
      <organizations>
        <organization>North Inlet LTER</organization>
        <organization>NIN-LTER</organization>
      </organizations>
      <singledates></singledates>
      <timescales></timescales>
    </document>
  </resultset>
```

Search results XML (2 of 2)

```
    <site>nin</site>
    <taxonomic/>
    <title>
      Daily Water Sample Nutrient Data for North Inlet Estuary, South Carolina, from 1978 to 1992, North Inlet LTER
    </title>
    <authors>
      <author>Vernberg, F. John</author>
      <author>Blood, Elizabeth</author>
    </authors>
    <spatialCoverage>
      <coordinates>-79.2936 33.1925 -79.1002 33.357</coordinates>
    </spatialCoverage>
    <sources></sources>
    <keywords>
      <keyword>nutrient dynamics</keyword>
      <keyword>North Inlet Estuary</keyword>
      <keyword>Baruch Institute</keyword>
      <keyword>Georgetown, South Carolina</keyword>
    </keywords>
    <organizations>
      <organization>North Inlet LTER</organization>
      <organization>NIN-LTER</organization>
    </organizations>
    <singledates></singledates>
    <timescales></timescales>
  </document>
</resultset>
```

Search results XML (2 of 2)

```
    <site>nin</site>
    <taxonomic/>
    <title>
      Daily Water Sample Nutrient Data for North Inlet Estuary, South Carolina, from 1978 to 1992, North Inlet LTER
    </title>
    <authors>
      <author>Vernberg, F. John</author>
      <author>Blood, Elizabeth</author>
    </authors>
    <spatialCoverage>
      <coordinates>-79.2936 33.1925 -79.1002 33.357</coordinates>
    </spatialCoverage>
    <sources></sources>
    <keywords>
      <keyword>nutrient dynamics</keyword>
      <keyword>North Inlet Estuary</keyword>
      <keyword>Baruch Institute</keyword>
      <keyword>Georgetown, South Carolina</keyword>
    </keywords>
    <organizations>
      <organization>North Inlet LTER</organization>
      <organization>NIN-LTER</organization>
    </organizations>
    <singledates></singledates>
    <timescales></timescales>
  </document>
</resultset>
```

PASTA+ Search API: Three Simple Steps to Understanding

1. Indexing. What metadata fields can I query?
2. Querying. How do I compose a query and send it to PASTA+?
3. Search Results. What does PASTA+ send back to me?

Design and Implementation Decisions for a Local Data Catalog

- The user interface -- simple list or support for user query?
- Which programming language or CMS to interact with PASTA+ web services?
- How to transform the XML search results to HTML?
- How to construct links back to the data

Design and Implementation Decisions for a Local Data Catalog

- The interface
  - Just a simple listing of the full data catalog, or does it offer a more dynamic interface where users can enter a query to view a subset of your data catalog?
  - If you incorporate user search, is it a simple search box, or something more complex like a browse tree, a listbox with keywords to choose from, etc.?
  - Example query that combines user input with site filtering:

    https://pasta.lternet.edu/package/search/eml?q=keyword:Pocillopora&fq=scope:knb-lter-mcr&fl=*&rows=300

Design and Implementation Decisions for a Local Data Catalog

- Which programming language to interact with PASTA+ web services?
  - Virtually all modern programming languages and CMSs offer a means to interact with a web services API as a client. Anything that can pull content with an HTTP GET can use the PASTA+ Search API. Some of the typical choices in our community: Java, Python, PHP, R, content management systems (Drupal, WordPress), JavaScript
  - JavaScript is a particularly interesting choice because it can run in the browser as a static HTML file without a back-end web server.
    - Requires that the web services API supports something called CORS (Cross Origin Resource Sharing). PASTA+ supports CORS as of last week (Thanks to Dr. Tim Whiteaker, Information Manager at the Beaufort Lagoon Ecosystems LTER, for sharing his insights on JavaScript and CORS with the EDI technical team.)

Design and Implementation Decisions for a Local Data Catalog

- How to transform the XML search results to HTML?
  - XSLT is made for doing transformations, but a lot of people find it difficult to work with
  - Use the programming language to do the conversion: Python, R, etc.

Design and Implementation Decisions for a Local Data Catalog

- How to construct links back to the data

1. This PASTA+ Search API query:

   https://pasta.lternet.edu/package/search/eml?q=id:knb-lter-nin.1&fl=doi

2. Returns this XML:

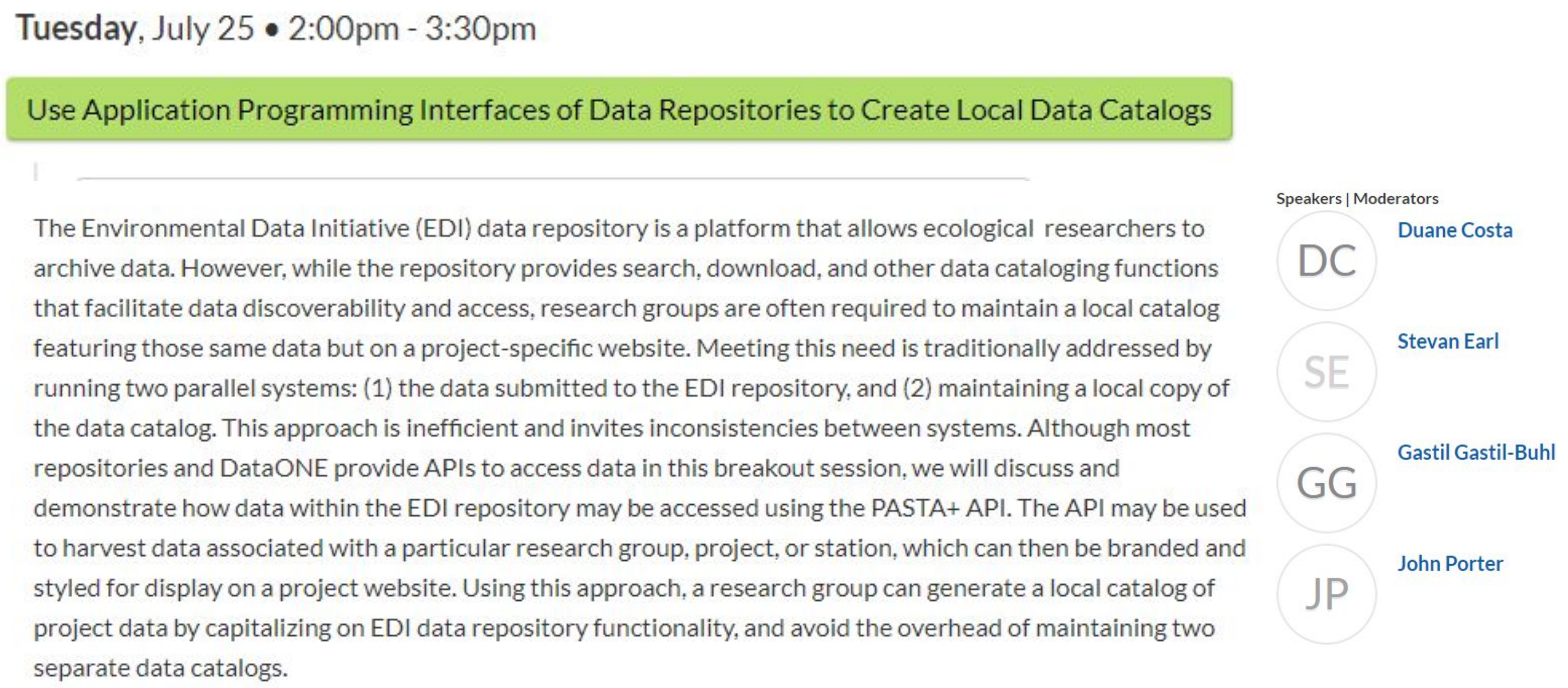
```
<resultset numFound="1" start="0" rows="10">
<document>
   <doi>doi:10.6073/pasta/0675d3602ff57f24838ca8d14d7f3961</doi>
  </document>
</resultset>
```

3. Which can then be used to formulate this link to the central data portal:
   https://doi.org/10.6073/pasta/0675d3602ff57f24838ca8d14d7f3961

4. Alternatively: Read the metadata from PASTA+. Parse and display whichever parts you choose in whatever style you choose:

   https://pasta.lternet.edu/package/metadata/eml/knb-lter-nin/1/newest

PASTA+ Search API: See Also ESIP 2017 Summer Meeting



Tuesday, July 25 • 2:00pm - 3:30pm

Use Application Programming Interfaces of Data Repositories to Create Local Data Catalogs

The Environmental Data Initiative (EDI) data repository is a platform that allows ecological researchers to archive data. However, while the repository provides search, download, and other data cataloging functions that facilitate data discoverability and access, research groups are often required to maintain a local catalog featuring those same data but on a project-specific website. Meeting this need is traditionally addressed by running two parallel systems: (1) the data submitted to the EDI repository, and (2) maintaining a local copy of the data catalog. This approach is inefficient and invites inconsistencies between systems. Although most repositories and DataONE provide APIs to access data in this breakout session, we will discuss and demonstrate how data within the EDI repository may be accessed using the PASTA+ API. The API may be used to harvest data associated with a particular research group, project, or station, which can then be branded and styled for display on a project website. Using this approach, a research group can generate a local catalog of project data by capitalizing on EDI data repository functionality, and avoid the overhead of maintaining two separate data catalogs.

Speakers | Moderators

- DC Duane Costa
- SE Stevan Earl
- GG Gastil Gastil-Buhl
- JP John Porter

PASTA+ Search API: See Also

- Using the PASTA+ Search API

  https://www.youtube.com/watch?v=IL7IsXGbFAE

- Use Application Programming Interfaces of Data Repositories to Create Local Data Catalogs

  https://www.youtube.com/watch?v=6hyKUKuxW54

- From the EDI Website (https://environmentaldatainitiative.org/) :
  - Resources → EDI on YouTube → Videos

Resources

PASTA+ Documentation - http://pastaplus-core.readthedocs.io

Browse & DIscovery - http://pastaplus-core.readthedocs.io/en/latest/doc_tree/pasta_api/data_package_manager_api.html#browse-and-discovery

PASTA+ Data Package API - https://pasta.lternet.edu/package/docs/api

Search API - https://pasta.lternet.edu/package/docs/api#GET : /search/eml

EDI Website - https://environmentaldatainitiative.org

EDI General Questions - info@environmentaldatainitiative.org

EDI Technical Issues - support@environmentaldatainitiative.org

JavaScript solution, Tim Whiteaker - https://github.com/twhiteaker/Solr-JavaScript-Search-Client

Thank you, and happy searching!