# NBN Atlas Query Primer

The search functionality of the NBN Atlas API is built from Apache's Lucene information retrieval library (https://en.wikipedia.org/wiki/Apache_Lucene) and sits on top of Apache's SOLR search platform (https://en.wikipedia.org/wiki/Apache_Solr). Many developers wishing to use the NBN Atlas API will already be familiar with this technology. Whilst this primer is directed primarily at those who are not already knowledgeable about SOLR, there are some implementation specifics which may still be helpful to those who are. Although the examples given refer to the species url, the API's other search queries support the same capabilities. Suggestions and corrections are gratefully received.

## Introduction to Facets and Query Filters

It is not recommended to use the main query parameter unless a fuzzy text ranking feature is specifically required (http://www.openjems.com/solr-lucene-score-tutorial/). It will usually be better to use a catch-all query (q=) and filter the results. Not only is this easier to debug (debugQuery is not available), the results are also cached and do not involve the calculation of a rank, and this may result in an increase in performance.

A catch-all query will notionally return the entire database[1]:
https://species-ws.nbnatlas.org/search.json?q=

A list of the available index fields is available at the following link (note, it is not possible to restrict which index fields are return by using SOLR's fl parameter, so all fields are always returned).
https://species-ws.nbnatlas.org/admin/indexFields

To discover what are commonly occurring values for a specific field (e.g. rk_kingdom) a facets request can be added to the original catch-all query (note, the pageSize parameter is set to 0 here to prevent any of the actual records being returned):
https://species-ws.nbnatlas.org/search.json?q=&facets=rk_kingdom&pageSize=0

This will return the range of values that the rk_kingdom field has (up to a maximum of 100) and a count of how many items in the database have each specific value.

A query filter (e.g. qf=rk_kingdom:Animalia) can be added to restrict the search to a specific kingdom, and the most commonly occurring values (this time for the rk_phylum field) can be found.
https://species-ws.nbnatlas.org/search.json?q=&fq=rk_kingdom:Animalia&facets=rk_phylum&pageSize=0

Similarly, to list the top 100 orders in the phylum chordata:
https://species-ws.nbnatlas.org/search.json?q=&fq=rk_phylum:Chordata&facets=rk_order&pageSize=0

Finally, to see the actual species records for all the species of primates recorded in the database a filter that allows species records from the order Primates can be constructed:
https://species-ws.nbnatlas.org/search.json?q=&fq=rk_order:Primates&rank=species

[1] The results will be in JSON. To format them so they are easier to read you can copy and paste them into an online viewer (for example, http://jsonviewer.stack.hu/ or http://json.parser.online.fr/).

Duncan Rowland 13/4/2017

# Field Types: Stored and Text vs String

When the index field is listed as “stored”, then its value can be returned by a query (otherwise it can only be search for and is not included in the returned record).

When an index field type is listed as text then fuzzy matching is applied (the default field is “text” and is of type text and so should generally be avoided). Other field types include “string” which allowed more complex filters to be defined. Filters include wildcards (* or ? for multiple or single characters) and combinators.

https://species-ws.nbnatlas.org/search.json?q=&fq=scientificName:Hom*pien?

Note that if a field is of type text (e.g. commonName), then the fuzzy match that is performed in not case-sensitive. For example, the following returns matches that include “Portuguese man-of-war”:
https://species-ws.nbnatlas.org/search.json?q=&fq=commonName:Man

Where if a field is of type string (e.g. commonNameSingle), then the match is not fuzzy and is case-sensitive. For example, the following only returns Homo sapiens as a match:
https://species-ws.nbnatlas.org/search.json?q=&fq=commonNameSingle:Man

And the following returns nothing (because of case-sensitivity).
https://species-ws.nbnatlas.org/search.json?q=&fq=commonNameSingle:man

To match both an upper and lowercase H in a string field requires a combination of search terms.

# Combining Search Terms (1)

The combinatorial operators are: OR (the default, and as so is what a space is interpreted as), AND, NOT, and parentheses for precedence. It is especially important that fields are specified in the filter, and spaces are used carefully (the examples here use + instead of a space for clarity, note those these are converted to a space by the browser).

For example:

**commonNameSingle:Red+Fox** will be converted to **commonNameSingle:Red Fox** and parsed as **commonNameSingle:Red OR text:Fox** and so is a combination of a fuzzy search of all the text in the database for Fox with a string search on the commonNameSingle field.

`X OR Y` means the whole field must match either X OR Y
`X AND Y` means the whole field must match both X AND Y (so only makes sense with wildcards)

For example:

**commonNameSingle:”Red Fox”**
**commonNameSingle:Red\ Fox**
both mean the whole commonNameSingle field must exactly match the whole string “Red Fox”

**commonNameSingle:Red*+AND+commonNameSingle:*Fox**
**commonNameSingle:(Red*+AND+*Fox)**
both mean the whole commonNameSingle field must start with Red… and end with …Fox.

**commonNameSingle:Man+OR+commonNameSingle:*Fox**
**commonNameSingle:(Man+OR+*Fox)**
**commonNameSingle:(Man+*Fox)**
all three mean the whole commonNameSingle field must be either Man, or end with …Fox.
(Note, the ORs are optional here, as an unescaped space will default to an OR query).

———————————————————————————————————————

`X NOT Y` is also available, and as it is infix needs preceding by *:* to negate a filter element

For example:

**commonNameSingle:(Red*+AND+*:*+NOT+*Fox)**
**commonNameSingle:(Red*+NOT+*Fox)**
both would match commonNameSingle fields that start with Red… and do not end with …Fox.

**commonName:(man+NOT+war)**
would match commonName fields that contain man somewhere in the text, but not war (case-insensitive). So this query will not contain include a record for the Portuguese man-of-war.

———————————————————————————————————————

`-` can be prepended to a term to mean any returned result must not include it. Similarly,
`+` can be prepended to a term to mean any result must also include it.
Note that + must be unencoded as %2B or the browser will convert it to a space.

For example:

**commonName:(man+-war)**
is similar to the preceding and would match commonNames that contain man but not war

**commonName:(man+%2Bwar)**
would match commonNames that contain man and also contained war.

———————————————————————————————————————

Finally, it is also possible to filter for ranges of values, for example:

**guid:[NHMSYS0000376770+TO+NHMSYS0000376780]**
which would match guid values that occur alphabetically between the the two strings.

# Combining Search Terms (2)

An alternative to constructing a single compound query that ANDs together multiple terms is to use multiple fq (which are effectively ANDed together anyway), and it has been suggested that this offers performance gains.

e.g. a query generally in the form **fq=A AND B** could be rewritten as **fq=A & fq=B**

# Query vs Filter Query Caveat

It is usually advised that it is preferable to use fq filters as these do not involve the calculation of a rank and can be cached. However, there is an important distinction for the NBN Atlas. A q query might do additional processing that a fq might not do. A good example is when lsid is used to filter for a genus rather than a species.

Suppose you have genus A having species A.x and A.y. There may be occurrence records that are just identified to genus level whilst others are identified to species level. For a query Q=lsid:genus

q=Q  will return  A + A.x + A.y

q=*:* & fq=Q  will only return  A

In this case, the later solution should be avoided since it will only return occurrences identified at the genus level and not those identified at the species level.

# Paging Results

By default only 10 records are returned by a query (note, by default a maximum of 100 facets are returned and it is *generally* not possible to change this value). The PageSize parameter is used to set the maximum number of record to return, and the start parameter used to page through the entire set.

For example:
https://species-ws.nbnatlas.org/search.json?q=fish&pageSize=20&start=0
https://species-ws.nbnatlas.org/search.json?q=fish&pageSize=20&start=20

Duncan Rowland 13/4/2017