Building Customized Data Pipelines Using the Entrez Programming Utilities (eUtils)

Eric Sayers and David Wheeler

Introduction

The Entrez Programming Utilities (eUtils) are a set of seven server-side programs that provide a stable interface into the Entrez query and database system at the National Center for Biotechnology Information (NCBI). The eUtils use a fixed URL syntax that translates a standard set of input parameters into the values necessary for various NCBI software components to search for and retrieve the requested data. The eUtils are therefore the structured interface to the Entrez system, which currently includes 23 databases covering a variety of biomedical data, including nucleotide and protein sequences, gene records, three-dimensional molecular structures, and the biomedical literature.

To access these data, a piece of software first posts an eUtils URL to NCBI, then retrieves the results of this posting, after which it processes the data as required. The software can thus use any computer language that can send a URL to the eUtils server and interpret the XML response; examples of such languages are Perl, Python, Java, and C++. Combining eUtils components to form customized data pipelines within these applications is a powerful approach to data manipulation.

This guide first describes the general function and use of the eUtils and then outlines strategies for creating customized data pipelines with examples in Perl.

Links

eUtils course

eUtils-announce mailing list

eUtils help documents

Entrez search

Entrez database model

Entrez help

NLM Citation: Sayers E, Wheeler D. Building Customized Data Pipelines Using the Entrez Programming Utilities (eUtils). In: NCBI Short Courses [Internet]. Bethesda (MD): National Center for Biotechnology Information (US); 2004-.

Entrez tools

Two Things to Remember Before Using the eUtils

The eUtils Access Entrez Databases

The eUtils access the core search and retrieval engine of the Entrez system and, therefore, are only capable of retrieving data that are already in Entrez. Although the majority of data at NCBI is in Entrez, there are several datasets that exist outside of the Entrez system. Before beginning a project with the eUtils, check that the desired data can be found within an Entrez database.

The Entrez System Identifies Database Records Using UIDs

Each Entrez database refers to the data records within it by an integer ID called a UID. Examples of UIDs are GI numbers for Nucleotide and Protein, PMIDs for PubMed, or MMDB-IDs for Structure. The eUtils use UIDs for both data input and output, and thus it is often critical, especially for advanced data pipelines, to know how to find the UIDs associated with the desired data before beginning a project with the eUtils.

Understanding Entrez

The Entrez Engine: EGQuery, ESearch, and ESummary

The core of Entrez is an engine that performs two basic tasks for any Entrez database: 1) assemble a list of UIDs that match a text query, and 2) retrieve a brief summary record called a Document Summary (DocSum) for each UID. In Entrez, UIDs are always integers, and each refers to a unique record in a given Entrez database; the common names of the UIDS are listed. Document Summaries are a familiar sight in any Entrez Web search and are shown in the results display seen immediately after a search is executed.

These two basic tasks of the Entrez engine are performed by ESearch and ESummary. ESearch returns a list of UIDs that match a text query in a given Entrez database, and ESummary returns DocSums that match a list of input UIDs. EGQuery is a global version of ESearch that searches all Entrez databases simultaneously. Because these three eUtils perform the two core Entrez functions, they function well for all Entrez databases.

Entrez Databases: EInfo, EFetch*, and ELink

A growing number of databases, such as PubMed, Nucleotide, Protein, and Structure, use the core Entrez search and retrieval engine. EInfo provides detailed information about a given database, including lists of the indexing fields in the database and the available links to other Entrez databases. Each Entrez database includes two primary enhancements to the raw data records: 1) software for producing a variety of display formats appropriate to

the given database, and 2) each record may be linked to records in other Entrez databases via a list of associated UIDs.

The display format function is performed by EFetch, which generates formatted output for a list of input UIDs. For example, EFetch can produce abstracts from Entrez PubMed or FASTA format from Entrez Protein. The linked-records function is performed by ELink, which generates a list of UIDs in a specified Entrez database that are linked to a set of input UIDs. For example, ELink can find Entrez SNP records linked to records in Entrez Nucleotide, or Entrez Domain records linked to records in Entrez Protein.

***Note:** EFetch is currently supported only in the following databases: PubMed, PubMed Central, Journals, Nucleotide, Protein, Genome, Gene, SNP, PopSet, and Taxonomy.

Using the Entrez History Server: EPost *et al.*

A powerful feature of the Entrez system is that it can store retrieved sets of UIDs temporarily on the servers so that they can be combined subsequently or otherwise manipulated. The Entrez History server provides this service and is accessed on the Web using either the Preview/Index or History tabs on Entrez search pages. Each of the eUtils can also use the History server, which assigns each set of UIDs an integer label called a query key (&query_key) and an encoded server address called a Web environment (&WebEnv). EPost allows any list of Primary IDs (UIDs) to be uploaded to the History Server and returns the query key and Web environment. ESearch can also post its output set of UIDs to the History Server. The resulting query key and Web environment from either EPost or ESearch can then be used in place of a UID list in ESummary, EFetch, and ELink, which is very convenient when dealing with large datasets.

Guidelines for Constructing URLs

Special Characters

When constructing URLs for the eUtils, please use lowercase characters for all parameters except `&WebEnv`. There is no required order for the URL parameters in an eUtils URL, and null values or inappropriate parameters are ignored. Avoid placing spaces in the URLs, particularly in queries. If a space is required, use a plus sign (+) instead of a space:

- Incorrect: `&id=352, 25125, 234,` ...
- Correct: `&id=352,25125,234,`...
- Incorrect: `&term=biomol mrna[properties] AND mouse[organism]`
- Correct: `&term=biomol+mrna[properties]+AND+mouse[organism]`

Other special characters, such as the # symbol used in referring to a query key on the History server, should be represented by their URL encodings (%23 for #).

Identifying Your Queries

NCBI recommends that you use the `&tool` and `&email` parameters to identify all of your eUtils URLs. For `&tool`, choose a value that uniquely identifies your software. If your name is John Smith, use, for example, `&tool=johnsmithsoft`. If your email address is jsmith@hotmail.com, use `&email=jsmith@hotmail.com`. This email address is used only to inform the creator of the software of any problems. The NCBI does not use these addresses for mailing lists, although you can join the eUtils-announce mailing list if you wish.

The Seven eUtils in Brief

- **EInfo:** provides the number of records indexed in each field of a given database, the date of the last update of the database, and the available links from the database to other Entrez databases. [reference documentation]
- **EGQuery:** responds to a text query with the number of records matching the query in each Entrez database. [reference documentation]
- **ESearch:** responds to a text query with the list of UIDs matching the query in a given database, along with the term translations of the query. [reference documentation]
- **ESummary:** responds to a list of UIDs with the corresponding document summaries. [reference documentation]
- **EPost:** accepts a list of UIDs, stores the set on the History Server, and responds with the corresponding query key and Web environment. [reference documentation]
- **EFetch:** responds to a list of UIDs with the corresponding data records. [reference documentation]
- **ELink:** responds to a list of UIDs in a given database with either a list of related IDs in the same database or a list of linked IDs in another Entrez database. [reference documentation]

Syntax and Initial Parsing of Entrez Queries

Text search strings entered into the Entrez system are converted into Entrez queries with the following format:

term1[field1] **Op** *term2[field2]* **Op** *term3[field3]* **Op** ...

where the `terms` are search terms each limited to a particular Entrez `field` in square brackets, all combined using one of three Boolean operators: `Op = AND, OR, or NOT`. These Boolean operators must be typed in all capital letters.

Example: `human[organism] AND topoisomerase[protein name]`

Entrez initially splits the query into a series of items that were originally separated by spaces in the query; therefore it is critical that spaces separate each term and Boolean operator. If the query consists *only* of a list of UID numbers (unique identifiers) or

accession numbers, the Entrez system simply returns the corresponding records and no further parsing is performed. If the query contains any Boolean operators (`AND`, `OR`, or `NOT`), the query is split into the terms separated by these operators, and then each term is parsed independently. The results of these searches are then combined according to the Boolean operators. Further details about the parsing of Entrez queries are given in the Appendix.

A full account of how to search Entrez can be found in the Entrez Help Document. Additional information is available from Entrez Help.

Handling Large Datasets

Uploading Large UID Lists

When uploading a large list of UIDs using ESearch or EPost, or when using such a list as input to ESummary, EFetch, or ELink, it is a good idea to split the list into smaller batches of around 500 records. A series of URLs can then be posted to upload the entire set.

Downloading Large Datasets

When using ESummary or EFetch to download large datasets, it can be more efficient to use a series of URLs governed by the `retstart` and `retmax` parameters to download smaller batches of records. An example of doing this is given in Application 3: Retrieving Large Datasets.

Special Considerations When Using ELink

Preserving Record-to-Record Correspondence in Links

ELink can find links to not only one set of UIDs but also to multiple sets of UIDs simultaneously. This is very useful for preserving specific record-to-record links after the ELink call. To do this, use a separate `&id` parameter for each group of UIDs that requires a separate list of linked UIDs. Consider the following URLs:

URL 1: `elink.fcgi?`
`dbfrom=nucleotide&db=protein&id=41282244,41282247,40789264`

URL 2: `elink.fcgi?`
`dbfrom=nucleotide&db=protein&id=41282244&id=41282247&id=40789264`

URL 3: `elink.fcgi?`
`dbfrom=nucleotide&db=protein&Webenv=Webenv&query_key=key`

Both URLs 1 and 2 return the same protein GI numbers (41282245, 4759258, 40789265), but URL 1 returns them as a group without information about which nucleotide record is linked to which protein record. URL 2, on the other hand, returns three groups of links,

one for each `&id` parameter, preserving the nucleotide-to-protein links. URL 3 is functionally equivalent to URL 1 (assuming that the three GIs in the example are stored in that Web Enviroment).

ELink and the History Server

Although ELink can accept a stored set of UIDs from the History server as input, this eUtil cannot load its output onto the History server. The consequence of this is that the linked UIDs found by ELink must be parsed out of the XML output and then provided as input to another eUtil, either directly using the `&id` parameter or by using EPost to store them explicitly on the History server. Then they can be passed to other eUtils.

Combining eUtils Calls to Create Entrez Applications

The eUtils are useful when used by themselves in single URLs; however, their full potential is reached when successive eUtil URLs are combined to create a data pipeline. When used within such pipelines, the Entrez History server simplifies complex retrieval tasks by allowing easy data transfer between successive eUtil calls. Listed below are several examples of pipelines produced by combining eUtils, with the arrows representing the passing of `WebEnv` and `query_key` values from one eUtil to another. These pipelines are discussed in detail below.

Basic Pipelines

- Retrieving data records matching an Entrez query
  - ESearch → ESummary
  - ESearch → EFetch
- Retrieving data records matching a list of UIDs
  - EPost → ESummary
  - EPost → EFetch
- Finding IDs linked to records matching an Entrez query
  - ESearch → ELink
- Finding IDs linked to other UIDs
  - EPost → ELink

Advanced Pipelines

- Retrieving data records in database B linked to records in database A matching an Entrez query
  - ESearch → ELink → ESummary
  - ESearch → ELink → EFetch
- Retrieving data records from a subset of an ID list defined by an Entrez query
  - EPost → ESearch → ESummary
  - EPost → ESearch → EFetch
- Retrieving a subset of data records, defined by an Entrez query, from a set of records in database B linked to a list of UIDs in database A

- ELink → EPost → ESearch → ESummary
- ELink → EPost → ESearch → EFetch

Basic Pipelines

ESearch → ESummary/EFetch

Input: Entrez query

Output: DocSums (ESummary) or formatted data (EFetch) that match the Entrez query

Step 1. Use ESearch to find IDs that match an Entrez query and store them on the History server.

```
esearch.fcgi?db=database&term=query&usehistory=y
```

Step 2. Parse the Web Environment (`Webenv`) and query key (`key`) parameters from the XML ouput.

Step 3. Use ESummary or EFetch to retrieve records for the stored dataset.

```
esummary.fcgi?db=database&WebEnv=Webenv&query_key=key
```

```
efetch.fcgi?db=database&WebEnv=Webenv&query_key=key
```

EPost → ESummary/EFetch

Input: List of UIDs

Output: DocSums (ESummary) or formatted data (EFetch) that match the Entrez query

Step 1. Use EPost to store the IDs on the History server.

```
epost.fcgi?db=database&id=id_list
```

Step 2. Parse the Web Environment (`Webenv`) and query key (`key`) parameters from the XML ouput.

Step 3. Use ESummary or EFetch to retrieve records for the stored dataset.

```
esummary.fcgi?db=database&WebEnv=Webenv&query_key=key
```

```
efetch.fcgi?db=database&WebEnv=Webenv&query_key=key
```

ESearch → ELink

Input: Entrez query

Output: Primary IDs in database B that are linked to records in database A matching the Entrez query

Step 1. Use ESearch to find IDs that match an Entrez query and store them on the History server.

```
esearch.fcgi?db=databaseA&term=query&usehistory=y
```

Step 2. Parse the Web Environment (`Webenv`) and query key (`key`) parameters from the XML ouput.

Step 3. Use ELink to retrieve linked IDs for the stored dataset.

```
elink.fcgi?dbfrom=databaseA&db=databaseB&WebEnv=Webenv&query_key=key
```

EPost → ELink

Input: List of UIDs in database A

Output: List of UIDs in database B linked to the IDs in database A

Step 1. Use EPost to store the IDs on the History server.

```
epost.fcgi?db=databaseA&id=id_list
```

Step 2. Parse the Web Environment (`Webenv`) and query key (`key`) parameters from the XML ouput.

Step 3. Use ELink to retrieve linked IDs for the stored dataset.

```
elink.fcgi?dbfrom=databaseA&db=databaseB&WebEnv=Webenv&query_key=key
```

Advanced Pipelines

ESearch → ELink → ESummary/EFetch

Input: Entrez query

Output: DocSums (ESummary) or formatted data records (EFetch) in database B that are linked to records in database A matching the Entrez query

Step 1. Use ESearch to find IDs that match an Entrez query and store them on the History server.

```
esearch.fcgi?db=databaseA&term=query&usehistory=y
```

Step 2. Parse the Web Environment (`Webenv`) and query key (`key`) parameters from the XML ouput.

Step 3. Use ELink to retrieve linked IDs for the stored dataset.

```
elink.fcgi?dbfrom=databaseA&db=databaseB&WebEnv=Webenv&query_key=key
```

Step 4. Parse the UIDs from the ELink XML output and assemble as a comma-delimited list.

Step 5. Use ESummary or EFetch to retrieve data records corresponding to the ID list

```
esummary.fcgi?db=databaseB&id=id_list
```

```
efetch.fcgi?db=databaseB&id=id_list
```

EPost → ESearch → ESummary/EFetch

Input: List of UIDs

Output: DocSums (ESummary) or formatted data (EFetch) that correspond to the input list of IDs limited by an Entrez query

Step 1. Use EPost to store the IDs on the History server.

```
epost.fcgi?db=database&id=id_list
```

Step 2. Parse the Web Environment (`Webenv`) and query key (`key`) parameters from the XML ouput.

Step 3. Use ESearch to limit the stored dataset by an Entrez query.

```
esearch.fcgi?db=database&term=query+AND+%23key&WebEnv=Webenv&usehistory=y
```

Step 4. Parse the new Web Environment (`Webenv2`) and query key (`key2`) parameters from the XML ouput.

Step 5. Use ESummary or EFetch to retrieve records for the stored dataset.

```
esummary.fcgi?db=database&WebEnv=Webenv2&query_key=key2
```

```
efetch.fcgi?db=database&WebEnv=Webenv2&query_key=key2
```

ELink → EPost → ESearch → ESummary/EFetch

Input: List of UIDs

Output: DocSums (ESummary) or formatted data (EFetch) in database B that are both linked to input list of IDs in database A and match the Entrez query

Step 1. Use ELink to retrieve IDs in database B linked to IDs in database A.

```
elink.fcgi?dbfrom=databaseA&db=databaseB&id=id_list
```

Step 2. Parse the linked UIDs from the ELink XML output and assemble as a comma-delimited list (`id_list2`) for posting onto the History server.

Step 3. Use EPost to store the IDs on the History server.

```
epost.fcgi?db=databaseB&id=id_list2
```

Step 4. Parse the Web Environment (`Webenv`) and query key (`key`) parameters from the XML ouput.

Step 5. Use ESearch to limit the stored dataset by an Entrez query.

```
esearch.fcgi?db=databaseB&term=query+AND+%23key&WebEnv=Webenv&usehistory=y
```

Step 6. Parse the new Web Environment (`Webenv2`) and query key (`key2`) parameters from the XML ouput.

Step 7. Use ESummary or EFetch to retrieve records for the stored dataset.

```
esummary.fcgi?db=databaseB&WebEnv=Webenv2&query_key=key2
```

```
efetch.fcgi?db=databaseB&WebEnv=Webenv2&query_key=key2
```

Sample Applications of the eUtils

In the applications below, it is assumed that Perl is being used to create eUtils pipelines. In Perl, scalar variable names are preceded by a "$" symbol, and array names are preceded by a "@". In several instances, results will be stored in such variables for use in subsequent URLs.

Application 1: Converting GI Numbers to Accession Numbers

I have a list of nucleotide GI numbers and I want the corresponding accession numbers.

Solution: Use EFetch with `&rettype=acc`

URL: `efetch.fcgi?db=nucleotide&id=$gi_list&rettype=acc`

Application 2: Converting Accession Numbers to Data

I have a list of genome Accession numbers (`$acc_list`) and I want the sequences in FASTA format.

Solution: Use EFetch with `&rettype=fasta`

URL: `efetch.fcgi?db=genome&id=$acc_list&rettype=fasta`

Application 3: Retrieving Large Datasets

I want to retrieve an arbitrary number of formatted records that match an Entrez query.

Solution: First, run ESearch in Web Environment mode to retrieve the total number of UIDs that match the Entrez query (`<Count>` tag in the ESearch output). Then store this number into `$count`, and store the values of `WebEnv` and `query_key` into `$Webenv` and `$key`. Next, run EFetch multiple times, each time retrieving a batch of size `$retmax` (for example, `$retmax = 500`). Accomplish this by incrementing `$retstart` iteratively in a "for" loop to retrieve successive batches of records of size `$retmax`:

```
use LWP::Simple;
```

URL 1: `esearch.fcgi?db=database&term=$query&usehistory=y`

URL 2+: produced by the following loop:

Perl:

```
for ($retstart = 0; $retstart < $count; $restart += $retmax) {
 $efetch_url = $base ."db=$db&WebEnv=$Webenv&query_key=$key";
```

```
 $efetch_url .= "&retstart=$retstart&retmax=$retmax";
 $efetch_out = get($efetch_url);
 print "$efetch_out";
}
```

where `$base` = `http://eutils.ncbi.nlm.nih.gov/entrez/eutils/efetch.fcgi?`, `$db` is the database, and `$efetch_url` is a string containing the EFetch URL. This Perl code assumes that the `LWP::Simple` module is installed. This module allows the use of the get command for retrieving data from a URL.

Application 4: Downloading Contigs

I want to download a flatfile with the full sequence of an assembly (e.g., a contig).

Solution: Use EFetch with `&rettype=gbwithparts`

URL: `efetch.fcgi?db=nucleotide&id=27479347&rettype=gbwithparts`

Application 5: Limiting and Converting GI Lists

I have list of protein GI numbers from a BLAST search and I want to download the document summaries of only those protein records that are mammalian sequences with annotated SNPs.

Solution: Use EPost to upload the GI list, then use ESearch to limit the list, followed by EFetch to download the FASTA formatted data.

URL 1: `epost.fcgi?db=protein&id=$gi_list`

Result: In Perl, store WebEnv as `$Webenv1`, query_key as `$key1`

URL 2: `esearch.fcgi?db=protein&term=%23$key1+AND+mammalia[organism]+AND+protein+snp[filter]&usehistory=y&WebEnv=$Webenv1`

Result: In Perl, store WebEnv as `$Webenv2`, query_key as `$key2`

Note: The `%23` resolves to the # symbol, so that `%23$key1` → #2.

URL 3: `esummary.fcgi?db=protein&WebEnv=$Webenv2&query_key=$key2`

Application 6: Finding Related Records in Other Entrez Databases

I want to find all available 3D structure records similar to protein BAA20519.

Solution: Use ESearch to find the GI number, then ELink to find related sequences to that protein. Then use ELink again to find linked MMDB-IDs, and finally ESummary to download the document summaries of the structure records.

URL 1: `esearch.fcgi?db=protein&term=BAA20519`

Result: Find GI 2208903.

URL 2: `elink.fcgi?dbfrom=protein&db=protein&id=2208903`

Result: Find 1084 related sequences, extract into `$gi_list1`

URL 3: `elink.fcgi?dbfrom=protein&db=structure&id=$gi_list1`

Result: Find 9 related structures, extract into `$gi_list2`

URL 4: `esummary.fcgi?db=structure&id=$gi_list2`

Application 7: Entrez TBLASTX

I want to download all mRNAs from green plants that are related *at the protein level* to human NM_001126, in flatfile format.

Motivation: For finding distant homologs, protein BLAST searches are generally more sensitive than nucleotide BLAST searches. In this specific case, a nucleotide BLAST search finds no significant matches to NM_001126 from green plants, whereas TBLASTX will find several homologous sequences. However, TBLASTX is the most time-consuming version of BLAST, and therefore using the pre-computed results in Entrez saves significant computing time.

Solution: Use ESearch to retrieve the record for NM_001126, and then use ELink to find the linked protein sequence. Then use ELink again to find all related sequences to that protein, and then use ELink a third time to find all nucleotide records linked to those related proteins and then limit them to mRNAs from green plants. Finally, download the formatted data with EFetch.

URL 1: `esearch.fcgi?db=nucleotide&term=NM_001126`

Result: Find GI = 4557270.

URL 2: `elink.fcgi?dbfrom=nucleotide&db=protein&id=4557270`

Result: Find GI = 4557271.

URL 3: `elink.fcgi?dbfrom=protein&db=protein&id=4557271`

Result: Extract the 507 GI numbers into `$gi_list1`, and if desired, the raw BLAST scores reported by ELink into `@scores`

URL 4: `elink.fcgi?dbfrom=protein&db=nucleotide&id=$gi_list1&term=biomol+mrna[properties]+AND+viridiplantae[organism]`

Result: Extract the 7 GI numbers into `$gi_list2`

URL 5: `efetch.fcgi?db=nucleotide&WebEnv=$Webenv2&query_key=$key2&rettype=gb`

Result: Download the 7 plant mRNAs, none of which are found using Related Sequences to NM_001126

eUtils Course

The National Center for Biotechnology Information (NCBI) presents *NCBI PowerScripting*, a 3-day course that includes both lectures and computer workshops on using the NCBI eUtils effectively within scripts to automate search-and-retrieval operations across the entire suite of Entrez databases.

Full details of the course.

Appendix

Automatic Term Mapping in Entrez Queries

After the initial parsing, each resulting `term` in the Entrez query is then searched against three lists in the following order, and if a match is found, the indicated search is performed:

1. Taxonomic nodes → `taxonomic node[organism] OR term[All Fields]`
2. Journal names → `term[Journal]`
3. Author names → `term[Author]`

A valid author name is any word followed by a space and then one or two letters.

Subsequent Parsing

If no matches are found after automatic term mapping, the rightmost word of the term is removed, and automatic term mapping is repeated. This continues until either a match is found or the term is exhausted. If there is still no match, each word of the term is limited to *All Fields* and all terms are combined with `AND`.

Examples

- cancer cell receptor → `"Cancer Cell"[Journal] AND receptor[All Fields]`
- cell receptor cancer → `(cell[All Fields] AND receptor[All Fields]) AND ("Cancer"[Organism] OR cancer[All Fields])`
- human c-src kinase → `(("Homo sapiens"[Organism] OR human[All Fields]) AND c-src[All Fields]) AND kinase[All Fields]`
- wheat nuclear protein → `(("Triticum aestivum"[Organism] OR wheat[All Fields]) AND nuclear[All Fields]) AND protein[All Fields]`

- wheat w nuclear protein → `(wheat w[Author] AND nuclear[All Fields]) AND protein[All Fields]`