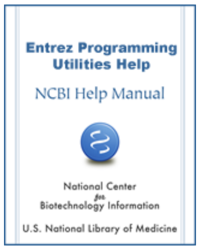
NLM Citation: Sayers E. Sample Applications of the E-utilities. 2009 Apr 24 [Updated 2017 Nov 1]. In: Entrez Programming Utilities Help [Internet]. Bethesda (MD): National Center for Biotechnology Information (US); 2010-.
Bookshelf URL: https://www.ncbi.nlm.nih.gov/books/



Sample Applications of the E-utilities

Eric Sayers, PhD[1]

Created: April 24, 2009; Updated: November 1, 2017.

Introduction

This chapter presents several examples of how the E-utilities can be used to build useful applications. These examples use Perl to create the E-utility pipelines, and assume that the LWP::Simple module is installed. This module includes the *get* function that supports HTTP GET requests. One example (Application 4) uses an HTTP POST request, and requires the LWP::UserAgent module. 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 E-utility calls. The code examples here are working programs that can be copied to a text editor and executed directly. Equivalent HTTP requests can be constructed in many modern programming languages; all that is required is the ability to create and post an HTTP request.

Basic Pipelines

All E-utility applications consist of a series of calls that we will refer to as a pipeline. The simplest E-utility pipelines consist of two calls, and any arbitrary pipeline can be assembled from these basic building blocks. Many of these pipelines conclude with either ESummary (to retrieve DocSums) or EFetch (to retrieve full records). The comments indicate those portions of the code that are required for either call.

ESearch – ESummary/EFetch

Input: Entrez text query

ESummary Output: XML Document Summaries

EFetch Output: Formatted data records (e.g. abstracts, FASTA)

```
use LWP::Simple;

# Download PubMed records that are indexed in MeSH for both asthma and
# leukotrienes and were also published in 2009.

$db = 'pubmed';
$query = 'asthma[mesh]+AND+leukotrienes[mesh]+AND+2009[pdat]';

#assemble the esearch URL
```

Author Affiliation: 1 NCBI; Email: sayers@ncbi.nlm.nih.gov.

Corresponding author.

```
$base = 'https://eutils.ncbi.nlm.nih.gov/entrez/eutils/';
$url = $base . "esearch.fcgi?db=$db&term=$query&usehistory=y";

#post the esearch URL
$output = get($url);

#parse WebEnv and QueryKey
$web = $1 if ($output =~ /<WebEnv>(\S+)<\/WebEnv>/);
$key = $1 if ($output =~ /<QueryKey>(\d+)<\/QueryKey>/);

### include this code for ESearch-ESummary
#assemble the esummary URL
$url = $base . "esummary.fcgi?db=$db&query_key=$key&WebEnv=$web";

#post the esummary URL
$docsums = get($url);
print "$docsums";

### include this code for ESearch-EFetch
#assemble the efetch URL
$url = $base . "efetch.fcgi?db=$db&query_key=$key&WebEnv=$web";
$url .= "&rettype=abstract&retmode=text";

#post the efetch URL
$data = get($url);
print "$data";
```

EPost – ESummary/EFetch

Input: List of Entrez UIDs (integer identifiers, e.g. PMID, GI, Gene ID)

ESummary Output: XML Document Summaries

EFetch Output: Formatted data records (e.g. abstracts, FASTA)

```
use LWP::Simple;

# Download protein records corresponding to a list of GI numbers.

$db = 'protein';
$id_list = '194680922,50978626,28558982,9507199,6678417';

#assemble the epost URL
$base = 'https://eutils.ncbi.nlm.nih.gov/entrez/eutils/';
$url = $base . "epost.fcgi?db=$db&id=$id_list";

#post the epost URL
$output = get($url);

#parse WebEnv and QueryKey
$web = $1 if ($output =~ /<WebEnv>(\S+)<\/WebEnv>/);
$key = $1 if ($output =~ /<QueryKey>(\d+)<\/QueryKey>/);

### include this code for EPost-ESummary
#assemble the esummary URL
$url = $base . "esummary.fcgi?db=$db&query_key=$key&WebEnv=$web";

#post the esummary URL
$docsums = get($url);
```

```
print "$docsums";

### include this code for EPost-EFetch
#assemble the efetch URL
$url = $base . "efetch.fcgi?db=$db&query_key=$key&WebEnv=$web";
$url .= "&rettype=fasta&retmode=text";

#post the efetch URL
$data = get($url);
print "$data";
```

Note: *To post a large number (more than a few hundred) UIDs in a single URL, please use the HTTP POST method for the EPost call (see Application 4).*

ELink – ESummary/Efetch

Input: List of Entrez UIDs in database A (integer identifiers, e.g. PMID, GI, Gene ID)

ESummary Output: Linked XML Document Summaries from database B

EFetch Output: Formatted data records (e.g. abstracts, FASTA) from database B

```
use LWP::Simple;

# Download gene records linked to a set of proteins corresponding to a list
# of GI numbers.

$db1 = 'protein';  # &dbfrom
$db2 = 'gene';     # &db
$linkname = 'protein_gene'; # desired link &linkname
#input UIDs in $db1 (protein GIs)
$id_list = '194680922,50978626,28558982,9507199,6678417';

#assemble the elink URL
$base = 'https://eutils.ncbi.nlm.nih.gov/entrez/eutils/';
$url = $base . "elink.fcgi?dbfrom=$db1&db=$db2&id=$id_list";
$url .= "&linkname=$linkname&cmd=neighbor_history";

#post the elink URL
$output = get($url);

#parse WebEnv and QueryKey
$web = $1 if ($output =~ /<WebEnv>(\S+)<\/WebEnv>/);
$key = $1 if ($output =~ /<QueryKey>(\d+)<\/QueryKey>/);

### include this code for ELink-ESummary
#assemble the esummary URL
$url = $base . "esummary.fcgi?db=$db&query_key=$key&WebEnv=$web";

#post the esummary URL
$docsums = get($url);
print "$docsums";

### include this code for ELink-EFetch
#assemble the efetch URL
$url = $base . "efetch.fcgi?db=$db2&query_key=$key&WebEnv=$web";
$url .= "&rettype=xml&retmode=xml";

#post the efetch URL
```

```
$data = get($url);
print "$data";
```

***Notes:** To submit a large number (more than a few hundred) UIDs to ELink in one URL, please use the HTTP POST method for the Elink call (see Application 4). The &linkname parameter is used to force ELink to return only one set of links (one &query_key) to simplify parsing. If more than one link is desired, the above code must be altered to parse the multiple &query_key values from the ELink XML output. This code uses ELink in "batch" mode, in that only one set of gene IDs is returned and the one-to-one correspondence between protein GIs and gene IDs is lost. To preserve this one-to-one correspondence, please see Application 4 below.*

ESearch – ELink – ESummary/EFetch

Input: Entrez text query in database A

ESummary Output: Linked XML Document Summaries from database B

EFetch Output: Formatted data records (e.g. abstracts, FASTA) from database B

```
use LWP::Simple;
# Download protein FASTA records linked to abstracts published
# in 2009 that are indexed in MeSH for both asthma and
# leukotrienes.

$db1 = 'pubmed';
$db2 = 'protein';
$linkname = 'pubmed_protein';
$query = 'asthma[mesh]+AND+leukotrienes[mesh]+AND+2009[pdat]';

#assemble the esearch URL
$base = 'https://eutils.ncbi.nlm.nih.gov/entrez/eutils/';
$url = $base . "esearch.fcgi?db=$db1&term=$query&usehistory=y";
#post the esearch URL
$output = get($url);

#parse WebEnv and QueryKey
$web1 = $1 if ($output =~ /<WebEnv>(\S+)<\/WebEnv>/);
$key1 = $1 if ($output =~ /<QueryKey>(\d+)<\/QueryKey>/);

#assemble the elink URL
$base = 'https://eutils.ncbi.nlm.nih.gov/entrez/eutils/';
$url = $base . "elink.fcgi?dbfrom=$db1&db=$db2";
$url .= "&query_key=$key1&WebEnv=$web1";
$url .= "&linkname=$linkname&cmd=neighbor_history";
print "$url\n";

#post the elink URL
$output = get($url);
print "$output\n";

#parse WebEnv and QueryKey
$web2 = $1 if ($output =~ /<WebEnv>(\S+)<\/WebEnv>/);
$key2 = $1 if ($output =~ /<QueryKey>(\d+)<\/QueryKey>/);

### include this code for ESearch-ELink-ESummary
#assemble the esummary URL
$url = $base . "esummary.fcgi?db=$db2&query_key=$key2&WebEnv=$web2";
#post the esummary URL
$docsums = get($url);
```

```
print "$docsums";

### include this code for ESearch-ELink-EFetch
#assemble the efetch URL
$url = $base . "efetch.fcgi?db=$db2&query_key=$key2&WebEnv=$web2";
$url .= "&rettype=fasta&retmode=text";
#post the efetch URL
$data = get($url);
print "$data";
```

Notes: *The &linkname parameter is used to force ELink to return only one set of links (one &query_key) to simplify parsing. If more than one link is desired, the above code must be altered to parse the multiple &query_key values from the ELink XML output. This code uses ELink in "batch" mode, in that only one set of PubMed IDs is returned and the one-to-one correspondence between PubMed IDs and their related PubMed IDs is lost. To preserve this one-to-one correspondence, please see Application 4 below.*

EPost – ELink – ESummary/EFetch

Input: List of Entrez UIDs (integer identifiers, e.g. PMID, GI, Gene ID) in database A

ESummary Output: Linked XML Document Summaries from database B

EFetch Output: Formatted data records (e.g. abstracts, FASTA) from database B

```
use LWP::Simple;

# Downloads gene records linked to a set of proteins corresponding
# to a list of protein GI numbers.

$db1 = 'protein';  # &dbfrom
$db2 = 'gene';     # &db
$linkname = 'protein_gene';
#input UIDs in $db1 (protein GIs)
$id_list = '194680922,50978626,28558982,9507199,6678417';

#assemble the epost URL
$base = 'https://eutils.ncbi.nlm.nih.gov/entrez/eutils/';
$url = $base . "epost.fcgi?db=$db1&id=$id_list";

#post the epost URL
$output = get($url);

#parse WebEnv and QueryKey
$web1 = $1 if ($output =~ /<WebEnv>(\S+)<\/WebEnv>/);
$key1 = $1 if ($output =~ /<QueryKey>(\d+)<\/QueryKey>/);

#assemble the elink URL
$base = 'https://eutils.ncbi.nlm.nih.gov/entrez/eutils/';
$url = $base . "elink.fcgi?dbfrom=$db1&db=$db2&query_key=$key1";
$url .= "&WebEnv=$web1&linkname=$linkname&cmd=neighbor_history";

#post the elink URL
$output = get($url);

#parse WebEnv and QueryKey
$web2 = $1 if ($output =~ /<WebEnv>(\S+)<\/WebEnv>/);
$key2 = $1 if ($output =~ /<QueryKey>(\d+)<\/QueryKey>/);
```

```
### include this code for ESearch-ELink-ESummary
#assemble the esummary URL
$url = $base . "esummary.fcgi?db=$db2&query_key=$key2&WebEnv=$web2";

#post the esummary URL
$docsums = get($url);
print "$docsums";

### include this code for ESearch-ELink-EFetch
#assemble the efetch URL
$url = $base . "efetch.fcgi?db=$db2&query_key=$key2&WebEnv=$web2";
$url .= "&rettype=xml&retmode=xml";

#post the efetch URL
$data = get($url);
print "$data";
```

Notes: *To post a large number (more than a few hundred) UIDs in a single URL, please use the HTTP POST method for the EPost call (see Application 4 below). The &linkname parameter is used to force ELink to return only one set of links (one &query_key) to simplify parsing. If more than one link is desired, the above code must be altered to parse the multiple &query_key values from the ELink XML output. This code uses ELink in "batch" mode, in that only one set of gene IDs is returned and the one-to-one correspondence between protein GIs and Gene IDs is lost. To preserve this one-to-one correspondence, please see Application 4 below.*

EPost – ESearch

Input: List of Entrez UIDs (integer identifiers, e.g. PMID, GI, Gene ID)

Output: History set consisting of the subset of posted UIDs that match an Entrez text query

```
use LWP::Simple;

# Given an input set of protein GI numbers, this script creates
# a history set containing the members of the input set that
# correspond to human proteins.
#(Which of these proteins are from human?)

$db = 'protein';
$query = 'human[orgn]';
$id_list = '194680922,50978626,28558982,9507199,6678417';

#assemble the epost URL
$base = 'https://eutils.ncbi.nlm.nih.gov/entrez/eutils/';
$url = $base . "epost.fcgi?db=$db&id=$id_list";

#post the epost URL
$output = get($url);

#parse WebEnv and QueryKey
$web = $1 if ($output =~ /<WebEnv>(\S+)<\/WebEnv>/);
$key = $1 if ($output =~ /<QueryKey>(\d+)<\/QueryKey>/);

#assemble the esearch URL
$term = "%23$key+AND+$query";
# %23 places a '#' before the query key
$url = $base . "esearch.fcgi?db=$db&term=$term";
$url .= "&WebEnv=$web&usehistory=y";
```

```
#post esearch URL
$limited = get($url);

print "$limited\n";

# Output remains on the history server (&query_key, &WebEnv)
# Use ESummary or EFetch as above to retrieve them
```

***Note:** To post a large number (more than a few hundred) UIDs in a single URL, please use the HTTP POST method for the EPost call (see Application 4).*

ELink – ESearch

Input: List of Entrez UIDs (integer identifiers, e.g. PMID, GI, Gene ID) in database A

Output: History set consisting of the subset of linked UIDs in database B that match an Entrez text query

```
use LWP::Simple;

# Given an input set of protein GI numbers, this script creates a
# history set containing the gene IDs linked to members of the input
# set that also are on human chromosome X.
#(Which of the input proteins are encoded by a gene on human
# chromosome X?)

$db1 = 'protein';  # &dbfrom
$db2 = 'gene';     # &db
$linkname = 'protein_gene'; # desired link &linkname
$query = 'human[orgn]+AND+x[chr]';
#input UIDs in $db1 (protein GIs)
$id_list = '148596974,42544182,187937179,4557377,6678417';

#assemble the elink URL
$base = 'https://eutils.ncbi.nlm.nih.gov/entrez/eutils/';
$url = $base . "elink.fcgi?dbfrom=$db1&db=$db2&id=$id_list";
$url .= "&linkname=$linkname&cmd=neighbor_history";

#post the elink URL
$output = get($url);

#parse WebEnv and QueryKey
$web = $1 if ($output =~ /<WebEnv>(\S+)<\/WebEnv>/);
$key = $1 if ($output =~ /<QueryKey>(\d+)<\/QueryKey>/);

#assemble the esearch URL
$term = "%23$key+AND+$query";  # %23 places a '#' before the query key
$url = $base . "esearch.fcgi?db=$db2&term=$term&WebEnv=$web&usehistory=y";

#post esearch URL
$limited = get($url);

print "$limited\n";

# Output remains on the history server (&query_key, &WebEnv)
# Use ESummary or EFetch as in previous examples to retrieve them
```

***Note:** To submit a large number (more than a few hundred) UIDs to ELink in one URL, please use the HTTP POST method for the Elink call (see Application 4). The &linkname parameter is used to force ELink to return only one set* *of links (one &query_key) to simplify parsing. If more than one link is desired, the above code must be altered to parse the multiple &query_key values from the ELink XML output. This code uses ELink in "batch" mode, in that only one set of gene IDs is returned and the one-to-one correspondence between protein GIs and Gene IDs is lost. To preserve this one-to-one correspondence, please see Application 4 below.*

Application 1: Converting GI numbers to accession numbers

Goal: Starting with a list of nucleotide GI numbers, prepare a set of corresponding accession numbers.

Solution: Use EFetch with &retttype=acc

Input: $gi_list – comma-delimited list of GI numbers

Output: List of accession numbers.

```
use LWP::Simple;
$gi_list = '24475906,224465210,50978625,9507198';

#assemble the URL
$base = 'https://eutils.ncbi.nlm.nih.gov/entrez/eutils/';
$url = $base . "efetch.fcgi?db=nucleotide&id=$gi_list&rettype=acc";

#post the URL
$output = get($url);
print "$output";
```

Notes: *The order of the accessions in the output will be the same order as the GI numbers in $gi_list.*

Application 2: Converting accession numbers to data

Goal: Starting with a list of protein accession numbers, return the sequences in FASTA format.

Solution: Create a string consisting of items separated by 'OR', where each item is an accession number followed by '[accn]'.

Example: accn1[accn]+OR+accn2[accn]+OR+accn3[accn]+OR+…

Submit this string as a &term in ESearch, then use EFetch to retrieve the FASTA data.

Input: $acc_list – comma-delimited list of accessions

Output: FASTA data

```
use LWP::Simple;
$acc_list = 'NM_009417,NM_000547,NM_001003009,NM_019353';
@acc_array = split(/,/, $acc_list);

#append [accn] field to each accession
for ($i=0; $i < @acc_array; $i++) {
   $acc_array[$i] .= "[accn]";
}

#join the accessions with OR
$query = join('+OR+',@acc_array);

#assemble the esearch URL
$base = 'https://eutils.ncbi.nlm.nih.gov/entrez/eutils/';
$url = $base . "esearch.fcgi?db=nuccore&term=$query&usehistory=y";
```

```
#post the esearch URL
$output = get($url);

#parse WebEnv and QueryKey
$web = $1 if ($output =~ /<WebEnv>(\S+)<\/WebEnv>/);
$key = $1 if ($output =~ /<QueryKey>(\d+)<\/QueryKey>/);

#assemble the efetch URL
$url = $base . "efetch.fcgi?db=nuccore&query_key=$key&WebEnv=$web";
$url .= "&rettype=fasta&retmode=text";

#post the efetch URL
$fasta = get($url);
print "$fasta";
```

Notes: *For large numbers of accessions, use HTTP POST to submit the esearch request (see Application 4), and see Application 3 below for downloading the large set in batches.*

Application 3: Retrieving large datasets

Goal: Download all chimpanzee mRNA sequences in FASTA format (>50,000 sequences).

Solution: First use ESearch to retrieve the GI numbers for these sequences and post them on the History server, then use multiple EFetch calls to retrieve the data in batches of 500.

Input: $query – chimpanzee[orgn]+AND+biomol+mrna[prop]

Output: A file named "chimp.fna" containing FASTA data.

```
use LWP::Simple;
$query = 'chimpanzee[orgn]+AND+biomol+mrna[prop]';

#assemble the esearch URL
$base = 'https://eutils.ncbi.nlm.nih.gov/entrez/eutils/';
$url = $base . "esearch.fcgi?db=nucleotide&term=$query&usehistory=y";

#post the esearch URL
$output = get($url);

#parse WebEnv, QueryKey and Count (# records retrieved)
$web = $1 if ($output =~ /<WebEnv>(\S+)<\/WebEnv>/);
$key = $1 if ($output =~ /<QueryKey>(\d+)<\/QueryKey>/);
$count = $1 if ($output =~ /<Count>(\d+)<\/Count>/);

#open output file for writing
open(OUT, ">chimp.fna") || die "Can't open file!\n";

#retrieve data in batches of 500
$retmax = 500;
for ($retstart = 0; $retstart < $count; $retstart += $retmax) {
        $efetch_url = $base ."efetch.fcgi?db=nucleotide&WebEnv=$web";
        $efetch_url .= "&query_key=$key&retstart=$retstart";
        $efetch_url .= "&retmax=$retmax&rettype=fasta&retmode=text";
        $efetch_out = get($efetch_url);
        print OUT "$efetch_out";
}
close OUT;
```

Application 4: Finding unique sets of linked records for each member of a large dataset

Goal: Download separately the SNP rs numbers (identifiers) for each current gene on human chromosome 20.

Solution: First use ESearch to retrieve the Gene IDs for the genes, and then assemble an ELink URL where each Gene ID is submitted as a separate &id parameter.

Input: $query – human[orgn]+AND+20[chr]+AND+alive[prop]

Output: A file named "snp_table" containing on each line the gene id followed by a colon (":") followed by a comma-delimited list of the linked SNP rs numbers.

```
use LWP::Simple;
use LWP::UserAgent;
$query = 'human[orgn]+AND+20[chr]+AND+alive[prop]';
$db1 = 'gene';
$db2 = 'snp';
$linkname = 'gene_snp';

#assemble the esearch URL
$base = 'https://eutils.ncbi.nlm.nih.gov/entrez/eutils/';
$url = $base . "esearch.fcgi?db=$db1&term=$query&usehistory=y&retmax=5000";

#post the esearch URL
$output = get($url);

#parse IDs retrieved
while ($output =~ /<Id>(\d+?)<\/Id>/sg) {
   push(@ids, $1);
}

#assemble  the elink URL as an HTTP POST call
$url = $base . "elink.fcgi";

$url_params = "dbfrom=$db1&db=$db2&linkname=$linkname";
foreach $id (@ids) {
   $url_params .= "&id=$id";
}

#create HTTP user agent
$ua = new LWP::UserAgent;
$ua->agent("elink/1.0 " . $ua->agent);

#create HTTP request object
$req = new HTTP::Request POST => "$url";
$req->content_type('application/x-www-form-urlencoded');
$req->content("$url_params");

#post the HTTP request
$response = $ua->request($req);
$output = $response->content;

open (OUT, ">snp_table") || die "Can't open file!\n";

while ($output =~ /<LinkSet>(.*?)<\/LinkSet>/sg) {
   $linkset = $1;
   if ($linkset =~ /<IdList>(.*?)<\/IdList>/sg) {
      $input = $1;
      $input_id = $1 if ($input =~ /<Id>(\d+)<\/Id>/sg);
   }

   while ($linkset =~ /<Link>(.*?)<\/Link>/sg) {
      $link = $1;
      push (@output, $1) if ($link =~ /<Id>(\d+)<\/Id>/);
   }

   print OUT "$input_id:" . join(',', @output) . "\n";

}

close OUT;
```

Notes: *This example uses an HTTP POST request for the elink call, as the number of Gene IDs is over 500. The &retmax parameter in the ESearch call is set to 5000, as this is a reasonable limit to the number of IDs to send to ELink in one request (if you send 5000 IDs, you are effectively performing 5000 ELink operations). If you need to link more than 5000 records, add &retstart to the ESearch call and repeat the entire procedure for each batch of 5000 IDs, incrementing &retstart for each batch.*

Demonstration Programs

Please see Chapter 1 for sample Perl scripts.

For More Information

Please see Chapter 1 for getting additional information about the E-utilities.