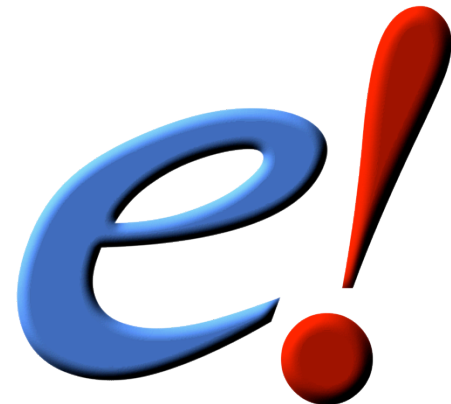


Ensembl API Course: REST API

Magali Ruffier
2nd September 2015



EMBL – European Bioinformatics Institute
Wellcome Trust Genome Campus
Hinxton, Cambridge, CB10 1SD, UK



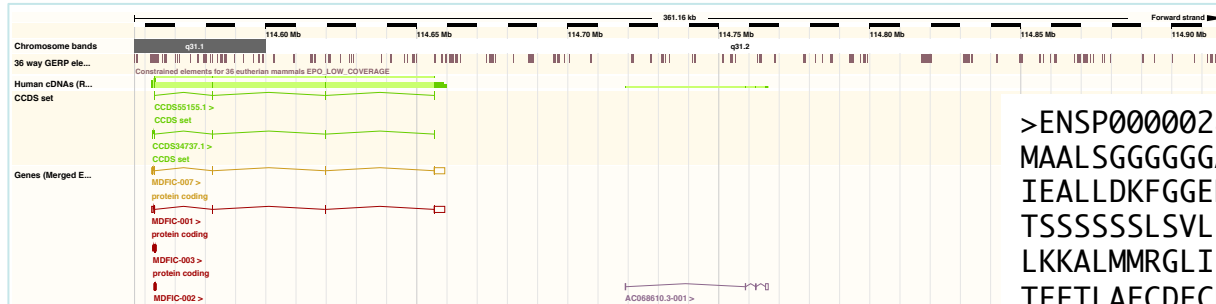
Outline

1. Ensembl
 1. Data available
 2. Data access
2. The REST API
 1. What is it
 2. How to query it
 3. Examples
 1. Retrieve sequence
 2. From region to variation
 3. Write a script

Documentation and Help

- <https://github.com/Ensembl/ensembl-presentation/>
 - This presentation
- <https://github.com/Ensembl/ensembl-rest/wiki>
 - How to write your first client
 - How to write POST requests
- dev@ensembl.org mailing list:
<http://www.ensembl.org/info/about/contact/mailling.html>
searchable mailing list archive:
<http://blog.gmane.org/gmane.science.biology.ensembl.devel>
- **Ensembl helpdesk:**
helpdesk@ensembl.org

Data in Ensembl



>ENSP00000288602

MAALSGGGGGAEPGQALFNGDMEPEAGAGAGAAASSAADPAIPEEV
IEALLDKFGGEHNPPSIYLEAYEEYTSKLDALQQREQQLLESLNGNT
TSSSSSSLSVLPSSLSVFQNPSTDVARSNPKSPQKPIVRVFLPNKQRT
LKKALMMRGLIPECCAVYRIQDGEKKPIGWDTDISWLTGEELHVEVL
TFFTALFCDFCRKLLFQGFRCQTCGYKFHQRCSTEVPLMCVNYDQLD
PQEEASLAETALTSGSSPSAPASDSIGPQILTSPSPSKSIPIQPFF
DRSSAPNVHINTIEPVNIDDLIRDQGFGRDGGSTTGLSATPPASLF
GPQREKSSSSSEDNRNMTLGRDSSDDWEIPDQGITVQGRIGSGS
AVKMLNVTAPTPQQLQAFKNEVGVLKTRHVNILLFMGYSTKPQLAI
LHIIETKFEMIKLIDIRQTAQGM DYLHAKSIIHRDLKSNNIFLHED
KSRWSGSHQFEQLSGSILWMAPEVIRMQDKNPYSFQSDVYAFGIVLY
NRDQIIFMVGRGYLSPDL SKVRSNCPKAMKRLMAECLKKKRDERPLF
LPKIHRSASEPSLNRA GFQTEDFSLYACASPKTPIQAGGYGAFPVH

rs111237862 SNP

Original source

Alleles

Location

Evidence status

Synonyms

HGVS names

Variants (including SNPs and indels) imported from dbSNP (release 137) | [View in dbSNP](#)

Reference/Alternative: **A/T** | Ancestral: **A** | Ambiguity code: **W**

Chromosome **7:114582393** (forward strand) | [View in location tab](#)



None currently in the database

This variation has **10** HGVS names - click the plus to show

Access to data

- Ensembl web site <http://www.ensembl.org>
- *Pre!* web site <http://pre.ensembl.org>
- *Archive!* web site <http://archive.ensembl.org>
- GRCh37 web site <http://grch37.ensembl.org>

- BioMart <http://www.ensembl.org/biomart/martview>

- FTP site <ftp://ftp.ensembl.org>
- MySQL <http://www.ensembl.org/info/data/mysql.html>
- Perl API <http://www.ensembl.org/info/data/api.html>
- REST API <http://rest.ensembl.org>
- GRCh37 REST API <http://grch37.rest.ensembl.org>

What Is A REST API

- **RE**presentational **S**tate **T**ransfer
- Simple APIs with very few external dependencies
- Can use the web (HTTP) to communicate
- Can be queried by
 - Web browsers
 - Command line tools
 - Programming languages

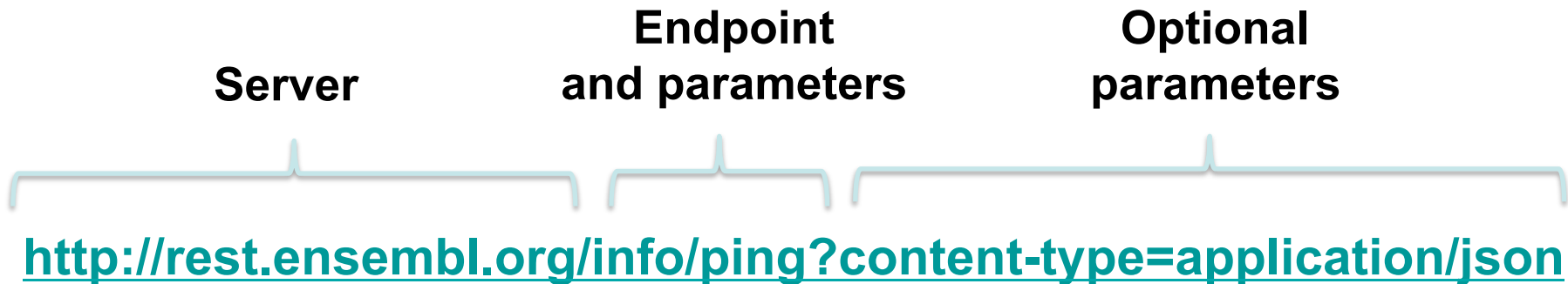


How Do You Query A REST API?

- With URLs; the same way you go to any webpage



How A URL Is Formed



How Do You Query?

- The web browser can be used as a test client
- Plugins are available for most browsers
 - Firefox – RESTClient
 - Chrome – REST Console or Postman
- All major programming languages can perform HTTP requests

Constructing a URL

1. Find the endpoint you need
 - Use find on the main page for more information
2. Copy an example URL into a text editor
3. Edit with your variables
 - Check the options available

4. Paste back into the address bar and press return

<http://rest.ensembl.org/sequence/region/:species/:region.fasta>

human

6:2198711..2198900

Example

Using the stable ID ENSG00000157764 (human BRAF)

- a) Retrieve its genomic sequence in plain text
- b) Soft mask this in FASTA format
- c) Retrieve all linked protein sequences in JSON

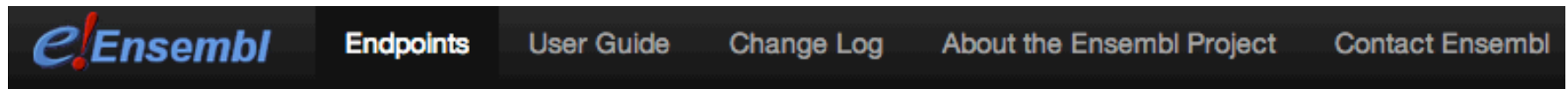
Example

Using the stable ID ENSG00000157764 (human BRAF)

- a) **Retrieve its genomic sequence in plain text**
- b) Soft mask this in FASTA format
- c) Retrieve all linked protein sequences in JSON

Start from the main page

<http://rest.ensembl.org>



Ensembl REST API Endpoints

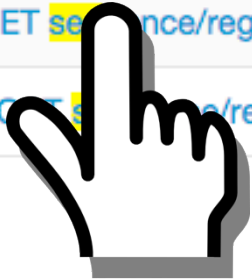
Archive

Resource	Description
GET archive/id/:id	Uses the given identifier to return the archived sequence
POST archive/id/	Retrieve the archived sequence for a set of identifiers

Find the correct endpoint

Sequence

Resource	Description
GET sequence/id/:id	Request multiple types of sequence by stable identifier.
POST sequence/id	Request multiple types of sequence by a stable identifier list.
GET sequence/region/:species/:region	Returns the genomic sequence of the specified region of the given species.
POST sequence/region/:species	Request multiple types of sequence by a list of regions.



Find and click [sequence/id/:id](#)

⌘ / Ctrl + f = find within page

Check the options available

The URL and required parameters

GET sequence/id/:id

Request multiple types of sequence by stable identifier.

Supported HTTP methods
and output formats

Parameters

Required

Name	Type	Description	Default	Example Values
id	String	An Ensembl stable ID	-	ENSG00000157764 ENSG00000157764.fasta (supported on some deployments)

Optional

Name	Type	Description	Default	Example Values
callback	String	Name of the callback subroutine to be returned by the requested JSONP response. Required ONLY when using JSONP as the serialisation method. Please see the user guide .	-	randomlygeneratedname
db_type	String	Restrict the search to a database other than the default. Useful if you need to use a DB other than core	-	core
expand_3prime	Int	Expand the sequence downstream of the sequence by this many basepairs. Only available when using genomic sequence type.	-	1000

Resource Information

Methods	GET
Response formats	fasta json text yaml jsonp
Slice length	1e7

The optional parameter set

Select an example

</sequence/id/ENSG00000157764?content-type=text/plain>

Example output

Perl

Python2

Python3

Ruby

Java

Curl

Wget

```
CGCCTCCCTTCCCCCTCCCCGCCCACAGCGGCCGCTCGGGCCCCGGCTCTCGGTTATAAGATGGCGGCGCTGAGCGGTGGCGGTGGTGGCGGCGCGGAGCCGGGCCAGGCTCTGTTCAACGGGGACA
TGGAGCCCAGAGCCGGCGCCGGCGCCGGCGCCGGCGCCCTCTCGGCTGCGGACCCTGCCATTCGGAGGAGGTGAGTGTGGCGCCACCCTGCCGCCCTCCGACTCCGGGCTCGGCGGCTGGCTGGTG
TTTATTTTGGAAAGAGGCGGCGGTGGGGGCTTGATGCCCTCAGCCACCTTCTCGGGCCAGCTCCGCGGGCTGGGAGGTGGGCATCGCCCCGTGTCCCTCTCCGTCATGCAGCGCCTTCTACGTAAAC
ACACACAATGGCCCCGGGGGTTTCCCTGGCCCCACCCAGATGTGGGGATTGGGGCAGCGGTGGTTGAGCGGGAGGCTATCAATAGGGGGCGAAACTCAGGGTTGGTCCGAGAAGGTCACGATTGGCT
GAAGTATCCAGCTCTGCATCTCTGTGGGGTGGGGGCGGCGGCGGCTCGACGTGGAGGATATAGGTTAGTTGCTGGGGCTGAGACAACAGCCCGAGTTACTGTCGCGTGTAAATCTTACATGGTCGTGG
GGATGATGGGGCTCATCTTCTCTCTCTCTCCCGACTGCCCCCTTCTCAGTCCGCTGCCCTTTTCACTTTTCTATTTGGGGATTCTCTTCACCTGTTTTACCCAGCAAATTATTTTGATTTA
GTCTTTACTTTTTCAATCCTAAATCGCAGTTTCCGATGCCTTTTCTGGTCTCTGGTCTCTGTTCTTAATGTTTGTGAGCGCTCTGTGCTGATTGGTAACCCCAATTCTATTCCCATCTACCGCCCGC
TCATTTTCCAGTTGTGCGACCTGCCTGCCTTCTAACCCAGCTCCCACTTAAGAGCATTTTTGCACTTCTCTTACCCTGGTCTCTTGAGGCTCTGACTTGATCTCACCCTCCCTAACATTGTTGTC
TGTTGTTATCTTCAAAATCCTCTGGACACTTTGGAGCTACTTGTTTTCTGAGCCAGAAGCTGTCAAGATTCCATCAGGTTTCACTTGGCTCTTTTCGCGCTTGCACTACTGGCACTTTTTGGCTAG
TCGTCCATTGTGATTACACCTCTTTATCTACCACTTTTATAGGCTGATTGATTTCTAGTGTTGCTCTCTTTTGTCTATTTTTTCTTTTCTTTTCTCTCCAGTCTTGCTCTCT
CAGCTGTTTTTGATTAGTCAGCTCTTAGCACTGTGTCAAATTATTTACGTTTTTTTATTACATAAAATTTATTACAAATATTTGGTATTTTATTACAGAAAATAATACTTTATTATGCTTTACAAA
TAAGATATGGTATAATAATTGTGGTTTACAGTTATTGATTAGGTAATGTGACTTACTCTGTTGACTTTGCTCGAAGTTCTCTTTGCTACTTACTATTAACATCTAATTTCTCAATTCTCATAACATCTC
ATTCTCTCTGCAATTTTTTTTTTGATCATCATCTTTGGAAATTCATCAATATGCTTGCTTTATTAGCATCAGCTTGTATGATAATGTTTGTCTTCTACTCTTTATATCATCTTTGTTACATGCC
CAAAATGTGTTCTGTACCATCATTTGATCTGTTCTAAAATTTCTCATTTTTAAGTTTCTTAAATCATTCCACTTTTCAGTATGCATTTTGTCTAGATCAGTTTCTCTCATATCTGTTCTTTCCCC
CAGCTTCTTGATTTCTAAGGAGAAAGCTCTTCTCTACTTCAATTTCTAGTTTATTCTGTTTCCCTGTTTCCAGTTACCATTCAATTTGCTTGTCTGCTGCTTTTGGTACTTAACTTTCTGAAGCT
TCCTCTTTTCTTCTCCACCTCCACGTTCTTCTTATTTATAACATCTTTGTTTCTTTGACATGGAAATTTATTTTAGGATACATTGTTTTAATGGATAAATACTAGGGGTCACATCTGCTGTC
TGTTTTCTCAGGAATCGGATATGCCTTTGTCTTAAACAGGCACAGGTGCCTCTGGATTTTATTTTACTCTGTAATAGATGTGTAGTTTTGTTGAATTTGATCTTGTGTTGAAGACTACTACAGAGTGA
```


Copy and paste in the navigation bar

← → ↻  rest.ensembl.org/sequence/id/ENSG00000157764?content-type=text/plain

CGCCTCCCTTCCCCCTCCCCGCCCCGACAGCGGCCGCTCGGGCCCCGGCTCTCGGTTATAAGATGGCGGCGCTGAGCGGTGGCGG
AAGAGGCGGCGGTGGGGGCTTGATGCCCTCAGCCACCTTCTCGGGCCAGCTCCGCGGGCTGGGAGGTGGGCATCGCCCCCGTGT
CTGTGGGGTGGGGGCGGCGGCGGCCCTCGACGTGGAGGATATAGGTTAGTTGCTGGGGCTGAGACAACAGCCCCGAGTTACTGTCC
TCCGATGCCTTTTCTGGTCTCTGGTCCTCTGTTCCCTAATGTTTGTCTAGCGCTCTGTCTGCTGATTGGTAACCCCCATTCTATTCC
CTTGTTTTCTGAGCCCAGAAGCTGTCAAGATTCCATCAGGTTTCACTTGGCTCTTTTTCGCGCTTGCACTACTGGCACTTTTTGC
GTTTTTTTATTACATAAAATTTATTACAAATATTTGGTATTTTATTACAGAAAATAATACTTTATTATGCTTTACAAATAAGA
TTATTCAGCATCAGCTTGTTTATGATAATGTTTGTTTTCTACTCTTTATATCATCTTTGTTACATGCCCAAATGTGTTCTGT
CCAGTTACCATTCATTTTGCCTTGTTTCCTGGCTTTTGGTACTTAACTTTCTGAAGCTTCCTCTTTTCTTCTCCACACCTCCAC
TGTAAGTTTTGTTGAATTGTATCTTGTTTGAAGACTACTACAGAGTGGAACAATGAGTGAAGTAATAAGTAGGGGTATGAAAT
GCCATGGATTTCTGTATTTGGCACATGTCTTGAGCAGTTCCCATGTACCAATCCTTGAGAACCCTCTAGGCTAGCTGAATTTAAC
GGAGGGTGACATTGATTAAAAAATGTATCTCTGAATGTAAATATCAGTATTACAGATGATAAAATAAATTCTCCAAGAAAT
AGGTTTTTTTTTTTTTAAATAAAAGTTTCCCAGAGGGAAATTTTCATCTAAAAAAAAGTCTGATTTCAAAGGGAAAGCAAGTCA

Example

Using the stable ID ENSG00000157764 (human BRAF)

- a) Retrieve its genomic sequence in plain text
- b) Soft mask this in FASTA format**
- c) Retrieve all linked protein sequences in JSON

Check the parameters

mask	<i>Enum(hard,soft)</i>	Request the sequence masked for repeat sequences. Hard will mask all repeats as N's and soft will mask repeats as lowercased characters. Only available when using genomic sequence type.	-	<i>hard</i>
------	------------------------	---	---	-------------

Check the formats available

Resource

Information

Methods	GET
Response formats	fasta json text yaml jsonp
Slice length	1e7

Output formats

- <https://github.com/Ensembl/ensembl-rest/wiki/Output-formats>

Format	Content-type	Extension	Notes
FASTA	text/x-fasta	.fasta	Sequence serialisation format. Only supported on the /sequence endpoint.
GFF3	text/x-gff3	.gff3	Genomic feature serialisation format. Only supported on the /overlap endpoint.
BED	text/x-bed	.bed	Browser Extensible Data format as defined by UCSC. Only supported on the /overlap endpoint.

Modify the required parameters

← → ↺  rest.ensembl.org/sequence/id/ENSG00000157764.fasta?mask=soft

```
>ENSG00000157764 chromosome:GRCh38:7:140719327:140924764:-1
cgccctcccttccccctccccgcgcgaCAGCGCCCGCTCGGGCCCCGGCTCTCGGTTATAA
GATGGCGGCGCTGAGCGGTGGCGGTGGTGGCGGCGCGGAGCCGGGCCAGGCTCTGTTCAA
CGGGGACATGGAGCCCCGAggcccggcgccggcgccggcgccggcgccgTCTTCGGCTGCGGA
CCCTGCCATTCGGGAGGAGGTGAGTGCTGGCGCCACCCTGCCGCCCTCCCGACTCCGGGC
TCGGCGGCTGGCTGGTGTATTATTTTGGAAAGAGGCGGCGGTGGGGGCTTGATGCCCTCAG
CCACCTTCTCGGGCCAGCTCCGCGGGCTGGGAGGTGGGCATCGCCCCCGTGTCCCTCTCC
GTCATGCAGCGCCTTCCTACGTAAACACACACAATGGCCCGGGGGGTTTCCCTGGCCCCC
ACCCCAGATGTGGGGATTGGGGCAGCGGTGGTTGAGCGGGAGGCTATCAATAGGGGGCGA
AACTCAGGGTTGGTCCGAGAAGGTCACGATTGGCTGAAGTATCCAGCTCTGCATCTCTGT
GGGGTGGGGGCGGCGGCGGCCTCGACGTGGAGGATATAGGTTAGTTGCTGGGGCTGAGAC
AACAGCCCCGAGTTACTGTCGCGTGAATTCTTACATGGTCGTGGGGATGATGGGGCTCAT
CATTTCTCTCTCTCTCTCCCGGACTGCCCCCCTTCTCAGTCCGCTGCCCTTTTTCACTTT
TCTATTTGGGGATTTCTCTTACCTGTTTTTACCCAGCAAATTATTTTGATTTAGTCTTTA
CTTTTTCAATCCTAAATCGCAGTTTCCGATGCCTTTTCTGGTCTCTGGTCCTCTGTTCCCT
AATGTTTGTGTCAGCGCTCTGTGCGCTGATTGGTAACCCCCATTCTATTCCCATCTACGCCC
GCTCATTTTCCAGTTGTGCGACCTGCCTGCCTTCTAACCCCAGCTCCCACTTAAGAGCAT
TTTTGCACTTCTCTTACCCTGGTCCTCTTGAGGCTCTGTACTTGATCTCACCCTCCCTA
```

Example





Using the stable ID ENSG00000157764 (human BRAF)

- a) Retrieve its genomic sequence in plain text
- b) Soft mask this in FASTA format
- c) Retrieve all linked protein sequences in JSON**

Check the parameters

type	<i>Enum(genomic,cds,cdna,protein)</i>	Type of sequence. Defaults to genomic where applicable, i.e. not translations. cdna refers to the spliced transcript sequence with UTR; cds refers to the spliced transcript sequence without UTR.	<i>genomic</i>	<i>cds</i>
------	---------------------------------------	--	----------------	------------

Modify the required parameters

← → ↻  rest.ensembl.org/sequence/id/ENSG00000157764?content-type=text/plain;type=protein ☆   

```
{"error": "Requesting a gene and type not equal to \"genomic\" can result in multiple sequences. 4 sequences detected. Please rerun your request and specify the multiple_sequences parameter"}
```

Check the parameters (again)

`multiple_sequences` *Boolean*

Allow the service to return more than 1 sequence per identifier. This is useful when querying for a gene but using a type such as protein.

0

-

Modify the required parameters

← → ↺ rest.ensembl.org/sequence/id/ENSG00000157764?type=protein;multiple_sequences=1

```
>ENSP00000419060
XSTTGLSATPPASLPGSLTNVKALQKSPGPQERERKSSSSSEDNRNMTLGRDSSDDWEI
PDGQITVGQRIGSGSFGTVYKKGWHDVAVKMLNVTAPTPQQLQAFKNEVGVLKTRHVN
ILLFMGYSTKPQLAIVTQWCEGSSLYHHLHIIETKFEMIKLIDARQTAQGMDYLHAKSI
IHRDLKSNNIFLHEDLTVKIGDFGLATVKSRSWSGSHQFEQLSGSILWMAPEVIRMQDKNP
YSFQSDVYAAGIVLYELMTGQLPYSNINNRDQIIFMVGRGYLSPDLSKVRSNCPKAMKRL
MAECLKKRDERPLFPQILASIELLARS LPKIHRSAEPLNRAGFQTEDFS LYACASPK
TPIQAGGYGEFAAFK
>ENSP00000288602
MAALSGGGGGGAEPGQALFNGDMEPEAGAGAGAAAASSAADPAIPEEVWNIKQMIKLTQEH
IEALLDKFGGEHNPPSIYLEAYEYTSKLDALQOREQQLES LGNGTDFSVSSSASMDTV
TSSSSSSLSVLPSSLVFNPTDVARSNPKSPQKPIVRVFLPNKQRTVVPARCGVTVRDS
LKKALMMRGLIPECCAVYRIQDGEKPIGWDTDISWLTGEELHVEVLENVPLTTHNFVRK
TFFTALFCDFCRKLLFQGFRCQTCGYKFHQRCSTEVPLMCVNYDQLDLLFVSKFFEHHPI
PQEEASLAETALTSGSSPSAPASDSIGPQILTSPSPSKSIPPIQPFPADEDHRNQFGQR
DRSSSAPNVHINTIEPVNIDDLIRDQGRGDGGSTTGLSATPPASLPGSLTNVKALQKSP
GPQERERKSSSSSEDNRNMTLGRDSSDDWEIPDGQITVGQRIGSGSFGTVYKKGWHDV
AVKMLNVTAPTPQQLQAFKNEVGVLKTRHVNILLFMGYSTKPQLAIVTQWCEGSSLYHH
LHIIETKFEMIKLIDARQTAQGMDYLHAKSI IHRDLKSNNIFLHEDLTVKIGDFGLATV
KSRSWSGSHQFEQLSGSILWMAPEVIRMQDKNPYSFQSDVYAAGIVLYELMTGQLPYSNIN
NRDQIIFMVGRGYLSPDLSKVRSNCPKAMKRLMAECLKKRDERPLFPQILASIELLARS
LPKIHRSAEPLNRAGFQTEDFS LYACASPKTPIQAGGYGAFPVH
>ENSP00000418033
IHRDLKSNNIFLHEDLTVKIGDFGLATVKSRSWSGSHQFEQLSGSILWMAPEVIRMQDKNP
YSFQSDVYAAGIVLYELMTGQLPYSNINNRDQVLCPPWEYNK
>ENSP00000420119
QALFNGDMEPEAGAGAGAAAASSAADPAIPEEVWNIKQMIKLTQEHIEALLDKFGGEHNPP
SIYLEAYEYTSKLDALQOREQQLES LGNGTDFSVSSSASMDTVTSSSSSSLSVLPSSL
SVFNPTDVARSNPKSPQKPIVRVFLPNKQRTVVPARCGVTVRDSLKKALMMRGLIPECC
AVYRIQDGSFLELT
```

Existing endpoints

Endpoint	Entry point	Output data
/alignment	A region	Whole genome alignments from Compara
/genetree	A gene or tree identifier	The corresponding gene tree
/homology	A gene	Corresponding orthologs and paralogs
/xrefs	An external symbol	Corresponding Ensembl object
/info	Species or nothing	Information about data available
/lookup	A stable identifier	Information related to that feature
/map	A region to convert	Equivalent coordinates in another context
/overlap	A region or feature	All features overlapping that region
/regulatory	A stable identifier	Corresponding regulatory features
/sequence	A stable identifier	The corresponding sequence
/variation	A variation id	Corresponding variation features
/vep	A region or a variation id	Corresponding variant consequences

Example

Starting with a region in GRCh37, 17:64216194-64218564:1

- a) Find the corresponding region in assembly GRCh38
- b) Find genes overlapping this region
- c) Explore the corresponding gene tree
- d) Look for variant consequences

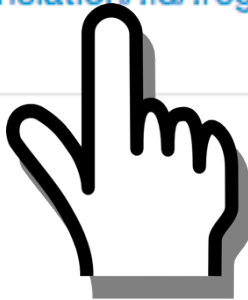
Example

Starting with a region in GRCh37, 17:64216194-64218564:1

- a) **Find the corresponding region in assembly GRCh38**
- b) Find genes overlapping this region
- c) Explore the corresponding gene tree
- d) Look for variant consequences

Find the correct endpoint

Resource	Description
GET map/cdna/:id/:region	Convert from cDNA coordinates to genomic coordinate API.
GET map/cds/:id/:region	Convert from CDS coordinates to genomic coordinates
GET map/:species/:asm_one/:region/:asm_two	Convert the co-ordinates of one assembly to another
GET map/translation/:id/:region	Convert from protein (translation) coordinates to genomic coordinates using the Ensembl API.



Select an example

Example Requests

</map/human/GRCh37/X:1000000..1000100:1/GRCh38?content-type=application/json>

Example output

Perl

Python2

Python3

Ruby

Java

Curl

Wget

```
{
  "mappings": [
    {
      "original": {
        "seq_region_name": "X",
        "strand": 1,
        "coord_system": "chromosome",
        "end": 1000100,
        "start": 1000000,
        "assembly": "GRCh37"
      },
      "mapped": {
        "seq_region_name": "X",
        "strand": 1,
        "coord_system": "chromosome",
        "end": 1039365,
        "start": 1039265
      }
    }
  ]
}
```


Copy and paste in the navigation bar

← → ↻  rest.ensembl.org/map/human/GRCh37/17:64216194-64218564:1/GRCh38

```
---
mappings:
  -
    mapped:
      assembly: GRCh38
      coord_system: chromosome
      end: 66222446
      seq_region_name: 17
      start: 66220076
      strand: 1
    original:
      assembly: GRCh37
      coord_system: chromosome
      end: 64218564
      seq_region_name: 17
      start: 64216194
      strand: 1
```

GRCh7:17:64216194-64218564:1 maps to GRCh38:17:66220076-66222446:1

Example

Starting with a region in GRCh37, 17:64216194-64218564:1

- a) Find the corresponding region in assembly GRCh38
- b) Find genes overlapping this region**
- c) Explore the corresponding gene tree
- d) Look for variant consequences

Find the correct endpoint

Resource	Description
GET overlap/id/:id	Retrieves features (e.g. genes, transcripts, variations et
GET overlap/region/:species/:region	Retrieves multiple types of features for a given region .
GET overlap/translation/:id	Retrieve features related to a specific Translation as de



Select an example

Example Requests

[/overlap/region/human/7:140424943-140624564?](#)

[feature=gene;feature=transcript;feature=cds;feature=exon;content-type=application/json](#)

Example output

[Perl](#)

[Python2](#)

[Python3](#)

[Ruby](#)

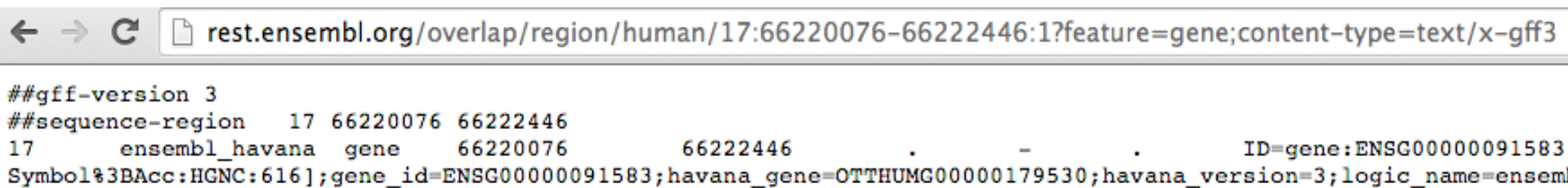
[Java](#)

[Curl](#)

[Wget](#)

```
[
  {
    "source": "ensembl_havana",
    "logic_name": "ensembl_havana_gene",
    "feature_type": "gene",
    "external_name": "RAB19",
    "havana_version": "1",
    "seq_region_name": "7",
    "havana_gene": "OTTHUMG00000157410",
    "strand": 1,
    "id": "ENSG00000146955",
```

Copy and paste in the navigation bar



The screenshot shows a web browser window with the address bar containing the URL: `rest.ensembl.org/overlap/region/human/17:66220076-66222446:1?feature=gene;content-type=text/x-gff3`. Below the address bar, the output of the API call is displayed in a monospaced font. The output starts with `##gff-version 3` and `##sequence-region 17 66220076 66222446`. The next line shows a gene feature: `17 ensembl_havana gene 66220076 66222446 . - . ID=gene:ENSG00000091583`. The final line is a detailed symbol description: `Symbol%3BACC:HGNC:616];gene_id=ENSG00000091583;havana_gene=OTTHUMG00000179530;havana_version=3;logic_name=ensem`.

```
##gff-version 3
##sequence-region 17 66220076 66222446
17 ensembl_havana gene 66220076 66222446 . - . ID=gene:ENSG00000091583
Symbol%3BACC:HGNC:616];gene_id=ENSG00000091583;havana_gene=OTTHUMG00000179530;havana_version=3;logic_name=ensem
```

Found one gene: ENSG00000091583

Example

Starting with a region in GRCh37, 17:64216194-64218564:1

- a) Find the corresponding region in assembly GRCh38
- b) Find genes overlapping this region
- c) Explore the corresponding gene tree**
- d) Look for variant consequences

Find the correct endpoint

Comparative Genomics

Resource	Description
GET genetree/id/:id	Retrieves a gene tree dump for a gene tree stable identifier
GET genetree/member/id/:id	Retrieves a gene tree that contains the stable identifier
GET genetree/member/symbol/:species/:symbol	Retrieves a gene tree containing the gene identified by a symbol



Select an example

Example Requests

</genetree/member/id/ENSG00000157764?content-type=text/x-phyloxml%2Bxml>

Example output

Perl

Python2

Python3

Ruby


Java

Curl

Wget

```
<?xml version="1.0" encoding="UTF-8"?><phyloxml xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://www.w.phyloxml.org" xsi:schemaLocation="http://www.phyloxml.org http://www.phyloxml.org/1.10/phyloxml.xsd">
  <phylogeny rooted="true" type="gene tree">
    <clade branch_length="0">
      <confidence type="duplication_confidence_score">1.0000</confidence>
      <taxonomy>
        <id>33213</id>
        <scientific_name>Bilateria</scientific_name>
      </taxonomy>
      <events>
```


Copy and paste in the navigation bar

← → ↻  rest.ensembl.org/genetree/member/id/ENSG00000091583

```
<?xml version="1.0" encoding="UTF-8"?>

<phyloxml xsi:schemaLocation="http://www.phyloxml.org http://www.phy
  <phylogeny rooted="true" type="gene tree">
    <clade branch_length="0">
      <confidence type="duplication_confidence_score">0.9853</confic
      <taxonomy>
        <id>33213</id>
        <scientific_name>Bilateria</scientific_name>
      </taxonomy>
      <events>
        <type>speciation_or_duplication</type>
        < duplications>1</duplications>
      </events>
    <clade branch_length="0">
      <taxonomy>
        <id>33213</id>
        <scientific_name>Bilateria</scientific_name>
      </taxonomy>
      <clade branch_length="0.014137">
        <confidence type="duplication_confidence_score">0.1970</cc
        <taxonomy>
          <id>7711</id>
```

Example

Starting with a region in GRCh37, 17:64216194-64218564:1

- a) Find the corresponding region in assembly GRCh38
- b) Find genes overlapping this region
- c) Explore the corresponding gene tree
- d) **Look for variant consequences**

Find the correct endpoint

VEP

Resource	Description
GET vep/:species/hgvs/:hgvs_notation	Fetch variant consequences based on a HGVS notation
GET vep/:species/id/:id	Fetch variation consequences based on a variation identifier
POST vep/:species/id	Fetch variation consequences for multiple ids
GET vep/:species/region/:region/:allele/	Fetch variation consequences
POST vep/:species/region	Fetch variation consequences for multiple regions

Variation



Resource	Description
GET variation/:species/	Uses a variation identifier (e.g. rsID) to return the variation features
POST variation/:species/	Uses a list of variation identifiers (e.g. rsID) to return the variation features

Select an example

Example Requests

</vep/human/region/9:22125503-22125502:1/C?content-type=application/json>

Example output

Perl

Python2

Python3

Ruby

Java


Curl

Wget

```
[
  {
    "assembly_name": "GRCh38",
    "end": 22125502,
    "seq_region_name": "9",
    "transcript_consequences": [
      {
        "gene_id": "ENSG00000240498",
        "distance": 4930,
        "variant_allele": "C",
        "biotype": "antisense",
        "gene_symbol_source": "HGNC",
        "consequence_terms": [
          "downstream_gene_variant"
        ],

```

Copy and paste in the navigation bar

← → ↺  rest.ensembl.org/vep/human/region/17:66220076-66222446:1/C?content-type=text/xml

This XML file does not appear to have any style information associated with it. The document tree is shown below.

```
▼<opt>
  ▼<data id="temp"
    allele_string="AAAGATTTTAAAATGTGTAACAATCAGACAGCAACCCAGTCTTAACCTTCATCTCCTAGGACTCTCCAATTTAACCCAGG"
    assembly_name="GRCh38" end="66222446" most_severe_consequence="splice_acceptor_variant" seq_region="17:66220076-66222446:1"
    ▼<transcript_consequences biotype="protein_coding" gene_id="ENSG00000091583" gene_symbol="APOH"
      <consequence_terms>splice_acceptor_variant</consequence_terms>
      <consequence_terms>splice_donor_variant</consequence_terms>
      <consequence_terms>coding_sequence_variant</consequence_terms>
      <consequence_terms>intron_variant</consequence_terms>
    </transcript_consequences>
    ▼<transcript_consequences biotype="protein_coding" gene_id="ENSG00000091583" gene_symbol="APOH"
      <consequence_terms>splice_acceptor_variant</consequence_terms>
      <consequence_terms>splice_donor_variant</consequence_terms>
      <consequence_terms>coding_sequence_variant</consequence_terms>
      <consequence_terms>intron_variant</consequence_terms>
    </transcript_consequences>
    ▼<transcript_consequences biotype="protein_coding" gene_id="ENSG00000091583" gene_symbol="APOH"
      <consequence_terms>splice_donor_variant</consequence_terms>
      <consequence_terms>coding_sequence_variant</consequence_terms>
    </transcript_consequences>
  </data>
</opt>
```

Write a script: documentation

Example output

Perl

Python2

Python3

Ruby

Java

Curl

Wget

Multiple stub clients designed to get you working with REST fast

```
8. my $server = 'http://rest.ensembl.org';
9. my $ext = '/overlap/region/human/7:140424943-140624564?feature=gene;feature=transcript;feature=cds;feature=exon';
10. my $response = $http->get($server.$ext, {
11.     headers => { 'Content-type' => 'application/json' }
12. });
13.
14. die "Failed!\n" unless $response->{success};
15.
16.
17. use JSON;
18. use Data::Dumper;
19. if(length $response->{content}) {
20.     my $hash = decode_json($response->{content});
21.     local $Data::Dumper::Terse = 1;
22.     local $Data::Dumper::Indent = 1;
23.     print Dumper $hash;
24.     print "\n";
25. }
26.
```

Show more advanced techniques e.g. providing headers and looking at server response codes for errors

Modify the example script

```
my $server = 'http://rest.ensembl.org';  
my $ext = '/overlap/region/human/7:140424943-140624564?feature=gene';  
my $response = $http->get($server.$ext, {  
    headers => { 'Content-type' => 'application/json' }  
});
```

Modify the example script

```
my $server = 'http://rest.ensembl.org';  
my $ext = '/overlap/region/human/7:140424943-140624564?feature=gene';  
my $response = $http->get($server.$ext, {  
    headers => { 'Content-type' => 'application/json' }  
});
```

```
my $server = 'http://rest.ensembl.org';  
my $region = '7:140424943-140624564';  
my $overlap_endpoint = '/overlap/region/human/';  
my $overlap_options = '?feature=gene';  
my $ext = $overlap_endpoint . $region . $overlap_options;  
my $response = $http->get($server.$ext, {  
    headers => { 'Content-type' => 'application/json' }  
});
```


Create re-usable methods

```
sub perform_rest_action {  
    my ($endpoint, $headers) = @_;  
    $parameters ||= {};  
    $headers ||= {};  
    $headers->{'Content-Type'} = 'application/json' unless exists $headers->{'Content-Type'};  
  
    my $url = $server.$endpoint;  
  
    my $response = $http->get($url, {headers => $headers});  
    my $status = $response->{status};  
    if(!$response->{success}) {  
        my ($status, $reason) = ($response->{status}, $response->{reason});  
        die "Failed for $endpoint! Status code: ${status}. Reason: ${reason}\n";  
    }  
}  
$request_count++;  
if(length $response->{content}) {  
    return $response->{content};  
}  
return;  
}
```

Multiple queries

/sequence/id

Example output

Example input

Perl

Python2

Python3

Ruby

Java

Curl

Wget

```
1. use strict;
2. use warnings;
3.
4. use HTTP::Tiny;
5.
6. my $http = HTTP::Tiny->new();
7.
8. my $server = 'http://rest.ensembl.org';
9. my $ext = '/sequence/id';
10. my $response = $http->request('POST', $server.$ext, {
11.     headers => {
12.         'Content-type' => 'application/json',
13.         'Accept' => 'application/json'
14.     },
15.     content => '{ "ids" : ["ENSG00000157764", "ENSG00000248378" ] }'
16. });
17.
18. die "Failed!\n" unless $response->{success};
19.
```

<https://github.com/Ensembl/ensembl-rest/wiki>

Ensembl REST 4.3 User Guide

The following guide refers to the 4.3 release of the Ensembl REST API. For support please contact [helpdesk](#) or our [dev mailing list](#).

Contents

1. [Writing Your First Client](#)
2. Example Clients (all query for a Gene and look for overlapping variation)
 - [Example Java Client](#)
 - [Example Perl Client](#)
 - [Example Python Client](#)
 - [Example Ruby Client](#)

Ensembl Acknowledgements

The Entire Ensembl Team

Fiona Cunningham¹, M. Ridwan Amode¹, Daniel Barrell^{1,2}, Kathryn Beal¹, Konstantinos Billis¹, Simon Brent², Denise Carvalho-Silva¹, Peter Clapham², Guy Coates², Stephen Fitzgerald¹, Laurent Gil¹, Carlos García Girón¹, Leo Gordon¹, Thibaut Hourlier¹, Sarah E. Hunt¹, Sophie H. Janacek¹, Nathan Johnson¹, Thomas Juettemann¹, Andreas K. Kähäri², Stephen Keenan¹, Fergal J. Martin¹, Thomas Maurel¹, William McLaren¹, Daniel N. Murphy^{1,2}, Rishi Nag¹, Bert Overduin¹, Anne Parker¹, Mateus Patricio¹, Emily Perry¹, Miguel Pignatelli¹, Harpreet Singh Riat¹, Daniel Sheppard¹, Kieron Taylor¹, Anja Thormann¹, Alessandro Vullo¹, Steven P. Wilder¹, Amonida Zadissa¹, Bronwen L. Aken¹, Ewan Birney¹, Jennifer Harrow², Rhoda Kinsella¹, Matthieu Muffato¹, Magali Ruffier¹, Stephen M.J. Searle², Giulietta Spudich¹, Stephen J. Trevanion¹, Andy Yates¹, Daniel R. Zerbino¹ and Paul Flicek^{1,2,*}

¹European Molecular Biology Laboratory, European Bioinformatics Institute, Wellcome Trust Genome Campus, Hinxton, Cambridge CB10 1SD, UK and ²Wellcome Trust Sanger Institute, Wellcome Trust Genome Campus, Hinxton, Cambridge, CB10 1SA, UK

Funding



Co-funded by the European Union



Standard Ensembl REST Parameters

:id – The global ID for that object. Normally a stable ID

ENSG00000157764

ENSGT003900000003602

:species – Species name (any Ensembl alias will do)

human

homo_sapiens

:region – 1 base locations CHR:START-END:STRAND

1:1000-2000

chr1:1000-2000:-1

:symbol – Reserved for Genes. Indicates a name to use

BRAF

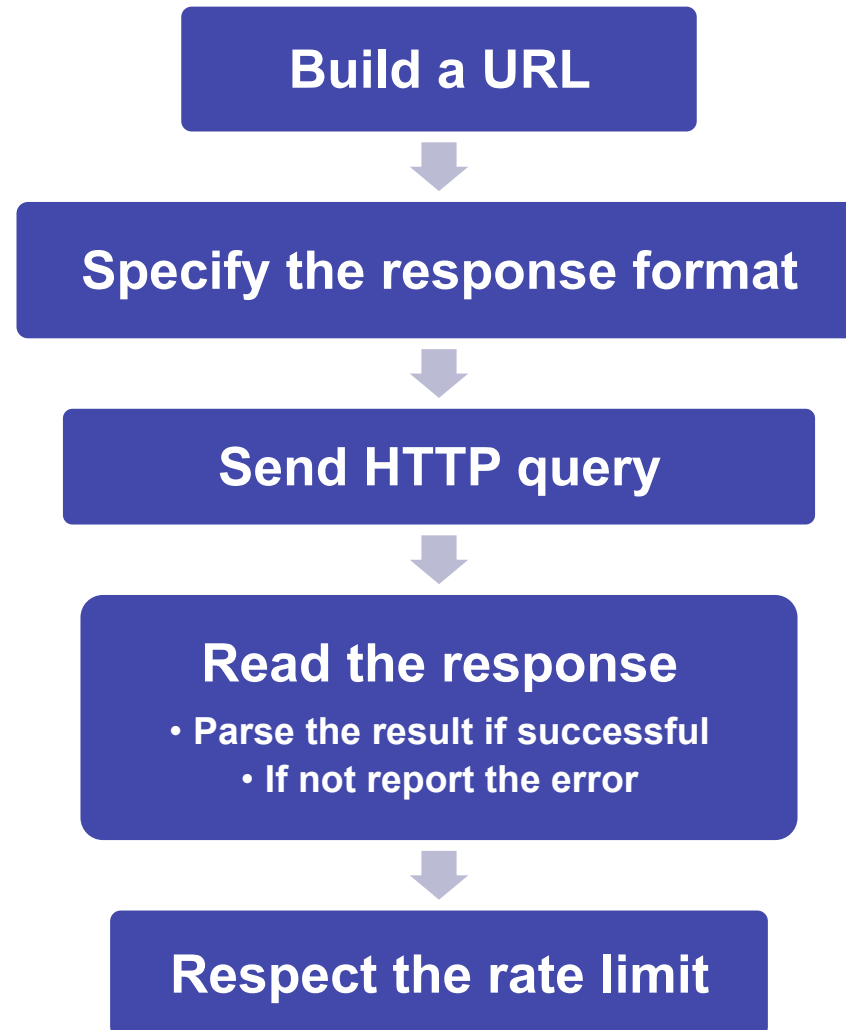
BRCA2

Species Required

Specifying Output Formats

- File extension
 - <http://rest.ensembl.org/sequence/id/ENST00000288602.json?type=cds>
- HTTP Parameter
 - <http://rest.ensembl.org/sequence/id/ENST00000288602?content-type=application/json&type=cds>
- HTTP Header
 - <http://rest.ensembl.org/sequence/id/ENST00000288602?type=cds>
 - **Content-Type: application/json**

Programming Against REST



Rules When Writing a Client

- Choose your most convenient language
 - If you are a Python programmer, use Python
- Find a good HTTP and JSON library for your language
- Try to limit your requests to 15 per second
 - We will limit you to ~54,000 per hour
 - Basic way is to count & sleep for fractions of seconds
- Create reusable methods for querying
 - It will make things easier

Pre-Written REST Clients

All of these are 3rd party contributed

- R - <https://github.com/acbb/EnsemblRest>
- Node.js - <https://github.com/jermth/EnsemblFetches>
- Ruby - <https://github.com/ALTree/bio-ensembl-rest>
- Java - <https://github.com/heuermh/ensembl-rest-client>
- Python - <https://github.com/pyOpenSci/pyEnsemblRest>

REST Terminology

Term	Definition	Example
Endpoint	A URL which will respond to a HTTP request and return data	/sequence/id/ ENST00000288602.fasta
HTTP Header	Sent to the server along side the requested URL.	Accepts: application/json Content-type: application/json
HTTP Parameter	Key value pairs separated by an = sign given to an endpoint after all required parameters.	?type=cds
HTTP Method	Tells the server the kind of operation you want to perform. Are you retrieving data or sending data to the server	GET POST
HTTP Status Code	Indicates to the client what the server did with a request ranging from “OK”, to “Server Error” and “Your Input Was Wrong”	200 400 404 500
JSON	Common cross-language data structure interchange format	{ “key”: [1, “two”]}

Output Formats

Format	Content-Type	Extension	Notes
FASTA	text/x-fasta	.fasta	
GFF3	text/x-gff3	.gff3	
JSON	application/json	.json	Standard serialisation format
JSONP	text/javascript	.jsonp	Used to avoid browser sandbox issues. Use CORS instead
Newick	text/x-nh	.nh	Old tree format. Use PhyloXML instead
SeqXML	text/x-seqxml+xml	.seqxml	FASTA replacement
PhyloXML	text/x-phyloxml+xml	.phyloxml	Phylogenetic format
Text	text/plain	.txt	
XML	text/xml	.xml	
YAML	text/x-yaml	.yaml	

Response Codes - Server Meta Data

Code	Class	Meaning
200	Success	Everything is groovy!
4xx	Client error	Range specifying some kind of error in the user request.
400	Client error	User has made a bad request e.g. bad parameters
404	Client error	Location cannot be found
415	Client error	Unsupported media type; bad format request made
429	Client error	Too many requests made. Observe the Retry-After header
5xx	Server error	Anything in the 500 range is a server issue. You cannot fix this
503	Server error	Server unavailable. Probably down for maintenance.

HTTP Response Headers

Name	Data type	Meaning
Content-type	MIME type	Describes what format the response is. The formats are the same as those used in the content-type request header
Retry-After	Floating point seconds	If found you must wait for this long before retrying the server
X-RateLimit-Limit	Integer	What the total limit of requests is
X-RateLimit-Remaining	Integer	How many requests you have left
X-RateLimit-Reset	Floating point seconds	How long before your tokens reset to the amount given in X-RateLimit-Limit
X-Runtime	Floating point seconds	The amount of time this request took

Documentation and Help

- <https://github.com/Ensembl/ensembl-rest/wiki>
 - How to write your first client
 - How to write POST requests
- dev@ensembl.org mailing list:
<http://www.ensembl.org/info/about/contact/mailling.html>
searchable mailing list archive:
<http://blog.gmane.org/gmane.science.biology.ensembl.devel>
- **Ensembl helpdesk:**
helpdesk@ensembl.org