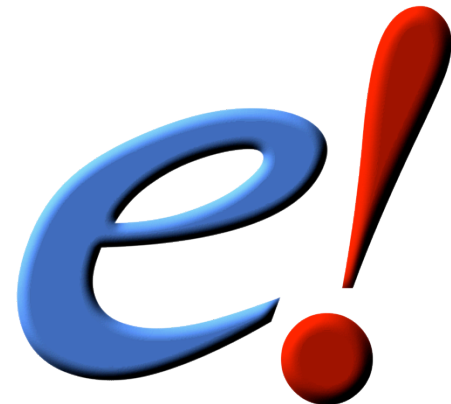


# Ensembl API Course: REST API

Magali Ruffier  
2<sup>nd</sup> September 2015



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



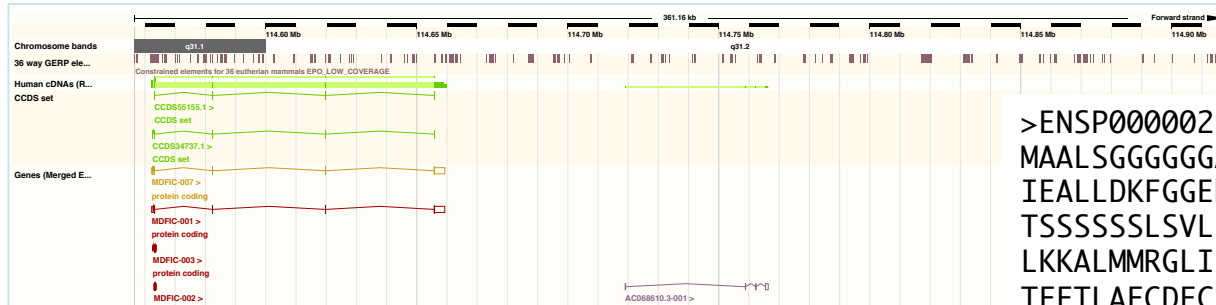
# Outline

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

# Presentation

- <https://github.com/Ensembl/ensembl-presentation/>
  - [https://github.com/Ensembl/ensembl-presentation/blob/master/API/REST/REST\\_bitesized.pptx?raw=true](https://github.com/Ensembl/ensembl-presentation/blob/master/API/REST/REST_bitesized.pptx?raw=true)
  - [https://github.com/Ensembl/ensembl-presentation/blob/master/API/REST/REST\\_bitesized.pdf?raw=true](https://github.com/Ensembl/ensembl-presentation/blob/master/API/REST/REST_bitesized.pdf?raw=true)

# Data in Ensembl



>ENSP00000288602

MAALSGGGGGGAEPGQALFNGDMEPEAGAGAGAAASSAADPAIPEEV  
IEALLDKFGGEHNPPSIYLEAYEEYTSKLDALQQREQQLLESLNGNT  
TSSSSSSLSVLPSSLSVFQNPSTDVARSNPKSPQKPIVRVFLPNKQRT  
LKKALMMRGLIPECCAVYRIQDGEKKPIGWDTDISWL TGEELHVEVL  
TFFTALFCDFCRKLLFQGFRCQTCGYKFHQRCSTEVPLMCVNYDQLD  
PQEEASLAETALTSGSSPSAPASDSIGPQILTSPSPSKSIPIQPFF  
DRSSAPNVHINTIEPVNIDDLIRDQGFGRDGGSTTGLSATPPASLF  
GPQREKSSSSSEDRNRMTLGRDSSDDWEIPDQGITVQGRIGSGS  
AVKMLNVTAPTPQQLQAFKNEVGVLKTRHVNILLFMGYSTKPLAI  
LHIIETKFEMIKLIDIARQTAQGM 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

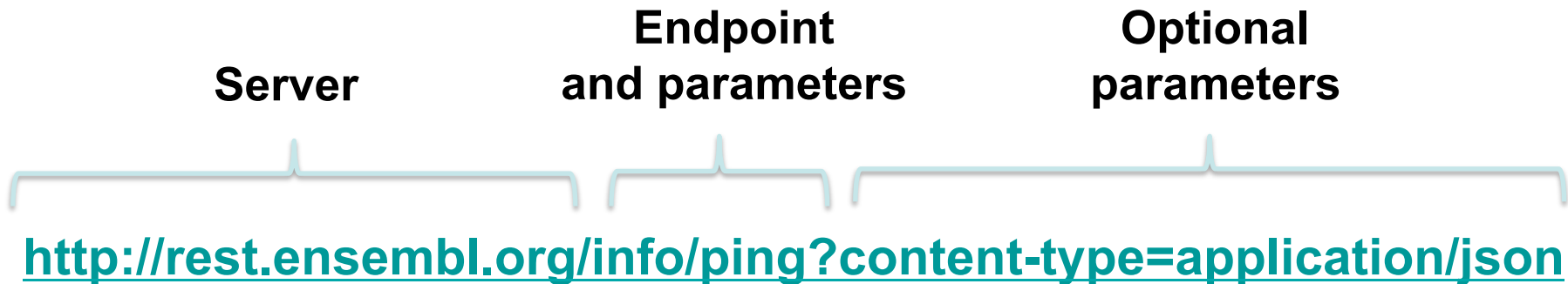


# How Do You Query A REST API?

- 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

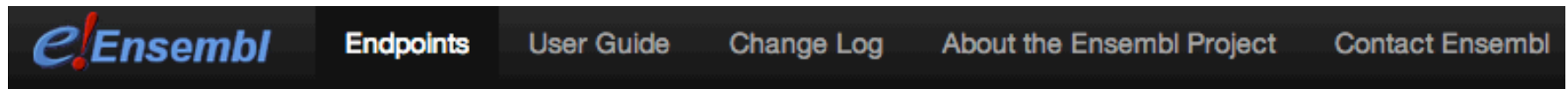
# Example

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

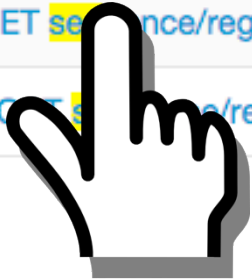
### Archive

Resource	Description
<a href="#">GET archive/id/:id</a>	Uses the given identifier to return the archived sequence
<a href="#">POST archive/id/</a>	Retrieve the archived sequence for a set of identifiers

# Find the correct endpoint

## Sequence

Resource	Description
GET <a href="#">sequence/id/:id</a>	Request multiple types of <b>sequence</b> by stable identifier.
POST <a href="#">sequence/id</a>	Request multiple types of <b>sequence</b> by a stable identifier list.
GET <a href="#">sequence/region/:species/:region</a>	Returns the genomic <b>sequence</b> of the specified region of the given species.
POST <a href="#">sequence/region/:species</a>	Request multiple types of <b>sequence</b> 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 <a href="#">the user guide</a> .	-	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
TGGAGCCCAGAGCCGGCGCCGGCGCCGGCGCCGGCCCTCTCGGCTGCGGACCCTGCCATTCGGAGGAGGTGAGTCTGGCGCCACCCTGCCGCCCTCCGACTCCGGGCTCGGCGGCTGGCTGGTG
TTTATTTTGGAAAGAGGCGGCGGTGGGGGCTTGATGCCCTCAGCCACCTTCTCGGGCCAGCTCCGCGGGCTGGGAGGTGGGCATCGCCCCGTGTCCCTCTCCGTCATGCAGCGCCTTCTACGTAAAC
ACACACAATGGCCCCGGGGGTTTCCCTGGCCCCACCCAGATGTGGGGATTGGGGCAGCGGTGGTTGAGCGGGAGGCTATCAATAGGGGGCGAAACTCAGGGTTGGTCCGAGAAGGTCACGATTGGCT
GAAGTATCCAGCTCTGCATCTCTGTGGGGTGGGGGCGGCGGCGGCTCGACGTGGAGGATATAGGTTAGTTGCTGGGGCTGAGACAACAGCCCGAGTTACTGTCGCGTGTAATTCTTACATGGTCGTGG
GGATGATGGGGCTCATCTTCTCTCTCTCTCCCGACTGCCCCCTTCTCAGTCCGCTGCCCTTTTCACTTTTCTATTTGGGGATTCTCTTCACCTGTTTTACCCAGCAAATTATTTTGATTTA
GTCTTTACTTTTTCAATCCTAAATCGCAGTTTCCGATGCCTTTTCTGGTCTCTGGTCTCTGTTCTTAATGTTTGTGAGCGCTCTGTGCTGATTGGTAACCCCAATTCTATTCCCATCTACCGCCCGC
TCATTTTCCAGTTGTGCGACCTGCCTGCCTTCTAACCCAGCTCCCACTTAAGAGCATTTTTGCACTTCTCTTACCCTGGTCTCTTGAGGCTCTGTACTTGATCTCACCCTCCCTAACATTGTTGTC
TGTTGTTATCTTCAAAATCCTCCTGGACACTTTGGAGCTACTTGTTTTCTGAGCCAGAAGCTGTCAAGATTCCATCAGGTTTCACTTGGCTCTTTTCGCGCTTGCACTACTGGCACTTTTTGGCTAG
TCGTCCATTGTGATTACACCTCTTTATTCTACCACTTTTATAGGCTGATTGATTTCTAGTGTTGCTCCTTTTGTCTATTTTTTCTTTTCTTTTCTCTCCAGTCTTGCTTCTCT
CAGCTGTTTTTGATTAGTCAAGCTCTTAGCACTGTGTCAAATTATTTACGTTTTTTTATTACATAAAATTTATTACAAATATTTGGTATTTTATTACAGAAAATAATACTTTATTATGCTTTACAAA
TAAGATATGGTATAATAATTGTGGTTTACAGTTATTGATTAGGTAATGTGACTTACTCTGTTGACTTTGCTCGAAGTTCTCTTTGCTACTTACTATTAACATCTAATTTCTCAATTCTCATAACATCTC
ATTCTCTCTGCAATTTTTTTTTTGATCATCATCTTTGGAAATTCATCAATATGCTTGCTTTATTAGCATCAGCTTGTTTATGATAATGTTTGTCTTCTACTCTTTATATCATCTTTGTTACATGCC
CAAAATGTGTTCTGTACCATCATTTGATCTGTTCTAAAATTTCTCATTTTTAAGTTTCTTAAATCATTCCACTTTTCAGTATGCATTTTGTCTAGATCAGTTTCTCTCATATCTGTTCTTTCCCC
CAGCTTCTTGATTTCTAAGGAGAAAGCTCTTCTCTACTTCAATTTCTAGTTTATTCTGTTTCCCTTGTTCAGTTACCATTCAATTTGCCTTGTTCCTGGCTTTTGGTACTTAACTTTCTGAAGCT
TCCTCTTTTCTTCTCCACCTCCACGTTCTTCTTATTTATAACATCTTTGTTTCTTTGACATGGAAATTTATTTTAGGATACATTGTTTTAATGGATAAATACTAGGGGTCACATCTGCTGTC
TGTTTTCTCAGGAATCGGATATGCCTTTGTCTTAAACAGGCACAGGTGCTCTGGATTTTATTTTACTCTGTAATAGATGTGTAGTTTTGTTGAATTGTATCTTGTGTTGAAGACTACTACAGAGTGA
```



# Copy and paste in the navigation bar

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

CGCCTCCCTTCCCCCTCCCCGCCCCGACAGCGGCCGCTCGGGCCCCGGCTCTCGGTTATAAGATGGCGGCGCTGAGCGGTGGCGG  
AAGAGGCGGCGGTGGGGGCTTGATGCCCTCAGCCACCTTCTCGGGCCAGCTCCGCGGGCTGGGAGGTGGGCATCGCCCCCGTGT  
CTGTGGGGTGGGGGCGGCGGCGGCCCTCGACGTGGAGGATATAGGTTAGTTGCTGGGGCTGAGACAACAGCCCCGAGTTACTGTCC  
TCCGATGCCTTTTCTGGTCTCTGGTCCTCTGTTCCCTAATGTTTGTTCAGCGCTCTGTCGCTGATTGGTAACCCCCATTCTATTCC  
CTTGTTTTTCTGAGCCCAGAAGCTGTCAAGATTCCATCAGGTTTCACTTGGCTCTTTTTCGCGCTTGCACTACTGGCACTTTTTGC  
GTTTTTTTATTACATAAAATTTATTACAAATATTTGGTATTTTATTACAGAAAATAATACTTTATTATGCTTTACAAATAAGA  
TTATTCAGCATCAGCTTGTTTATGATAATGTTTGTTTTCTACTCTTTATATCATCTTTGTTACATGCCCAAATGTGTTCTGT  
CCAGTTACCATTCATTTTGCCTTGTTTCCTGGCTTTTGGTACTTAACTTTCTGAAGCTTCCTCTTTTCTTCTCCACACCTCCAC  
TGTAGTTTTGTTGAATTGTATCTTGTTTGAAGACTACTACAGAGTGGAACAATGAGTGAAGTAATAAGTAGGGGTATGAAAT  
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](http://rest.ensembl.org/sequence/id/ENSG00000157764.fasta?mask=soft)

```
>ENSG00000157764 chromosome:GRCh38:7:140719327:140924764:-1
cgccctcccttccccctccccgccccgaCAGCGCCCGCTCGGGCCCCGGCTCTCGGTTATAA
GATGGCGGCGCTGAGCGGTGGCGGTGGTGGCGGCGCGGAGCCGGGCCAGGCTCTGTTCAA
CGGGGACATGGAGCCCCGAggcccggcgccggcgccggcgccggcgccgcggccTCTTCGGCTGCGGA
CCCTGCCATTCGGGAGGAGGTGAGTGCTGGCGCCACCCTGCCGCCCTCCCGACTCCGGGC
TCGGCGGCTGGCTGGTGTATTATTTTGGAAAGAGGCGGCGGTGGGGGCTTGATGCCCTCAG
CCACCTTCTCGGGCCAGCTCCGCGGGCTGGGAGGTGGGCATCGCCCCCGTGTCCCTCTCC
GTCATGCAGCGCCTTCCTACGTAAACACACACAATGGCCCCGGGGGGTTTCCCTGGCCCCC
ACCCCAGATGTGGGGATTGGGGCAGCGGTGGTTGAGCGGGAGGCTATCAATAGGGGGCGA
AACTCAGGGTTGGTCCGAGAAGGTCACGATTGGCTGAAGTATCCAGCTCTGCATCTCTGT
GGGGTGGGGGCGGCGGGCGGCCTCGACGTGGAGGATATAGGTTAGTTGCTGGGGCTGAGAC
AACAGCCCCGAGTTACTGTCGCGTGTAAATCCTTACATGGTTCGTGGGGATGATGGGGCTCAT
CATTTCTCTCTCTCTCTCCCGGACTGCCCCCCTTCTCAGTCCGCTGCCCTTTTTCACTTT
TCTATTTGGGGATTTCTCTTCACCTGTTTTTACCCAGCAAATTATTTTGATTTAGTCTTTA
CTTTTTCAATCCTAAATCGCAGTTTCCGATGCCTTTTCTGGTCTCTGGTCCTCTGTTCCCT
AATGTTTGTTCAGCGCTCTGTGCGCTGATTGGTAACCCCCATTCTATTCCCATCTACCGCCC
GCTCATTTTCCAGTTGTGCGACCTGCCTGCCTTCTAACCCCAGCTCCCACTTAAGAGCAT
TTTTGCACTTCTCTTACCCTGGTCCTCTTGAGGCTCTGTACTTGATCTCACCACCTCCCTA
```

# 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](http://rest.ensembl.org/sequence/id/ENSG00000157764?type=protein;multiple_sequences=1)

```
>ENSP00000419060
XSTTGLSATPPASLPGSLTNVKALQKSPGPQERERKSSSSSEDRNRMKTLGRRDSSDDWEI
PDGQITVGQRIGSGSFGTVYKKGWHDVAVKMLNVTAPTPQQLQAFKNEVGVLKTRHVN
ILLFMGYSTKPQLAIVTQWCEGSSLYHHLHIIETKFEMIKLIDARQTAQGMDYLHAKSI
IHRDLKSNNIFLHEDLTVKIGDFGLATVKSRWSGSHQFEQLSGSILWMAPEVIRMQDKNP
YSFQSDVYAAGIVLYELMTGQLPYSNINNRDQIIFMVGRGYLSPDLSKVRSNCPKAMKRL
MAECLKKRDERPLFPQILASIELLARS LPKIHRSAEPSLN RAGFQTEDFS LYACASPK
TPIQAGGYGEFAAFK
>ENSP00000288602
MAALSGGGGGGAEPGQALFNGDMEPEAGAGAGAAAASSAADPAIPEEVVNIQMIKLTQEH
IEALLDKFGGEHNPPSIYLEAYEYTSKLDALQOREQQLES LGNGTDFSVSSSASMDTV
TSSSSSSLSVLPSSLVVFQNP TDVARSNPKSPQKPIVRVFLPNKQRTVVPARCGVTVRDS
LKKALMMRGLIPECCAVYRIQDGEKPIGWDTDISWLTGEELHVEVLENVPLTTHNFVRK
TFFT LAFCDFCRKL LFQGFRCQTCGYKFHQRCSTEVPLMCVNYDQLDLLFVSKFFEHHPI
PQEEASLAETALTS GSSPSAPASDSIGPQILTSPSPSKSIPPIQPFPADEDHRNQFGQR
DRSSSAPNVHINTIEPVNIDDLIRDQGFGRDGGSTTGLSATPPASLPGSLTNVKALQKSP
GPQERERKSSSSSEDRNRMKTLGRRDSSDDWEIPDGQITVGQRIGSGSFGTVYKKGWHDV
AVKMLNVTAPTPQQLQAFKNEVGVLKTRHVNILLFMGYSTKPQLAIVTQWCEGSSLYHH
LHIIETKFEMIKLIDARQTAQGMDYLHAKSI IHRDLKSNNIFLHEDLTVKIGDFGLATV
KSRWSGSHQFEQLSGSILWMAPEVIRMQDKNPYSFQSDVYAAGIVLYELMTGQLPYSNIN
NRDQIIFMVGRGYLSPDLSKVRSNCPKAMKRLMAECLKKRDERPLFPQILASIELLARS
LPKIHRSAEPSLN RAGFQTEDFS LYACASPKTPIQAGGYGAFPVH
>ENSP00000418033
IHRDLKSNNIFLHEDLTVKIGDFGLATVKSRWSGSHQFEQLSGSILWMAPEVIRMQDKNP
YSFQSDVYAAGIVLYELMTGQLPYSNINNRDQVLCPPWEYNK
>ENSP00000420119
QALFNGDMEPEAGAGAGAAAASSAADPAIPEEVVNIQMIKLTQEHIEALLDKFGGEHNPP
SIYLEAYEYTSKLDALQOREQQLES LGNGTDFSVSSSASMDTVTSSSSSSLSVLPSSL
SVFQNP TDVARSNPKSPQKPIVRVFLPNKQRTVVPARCGVTVRDSLKKALMMRGLIPECC
AVYRIQDGSFLELT
```

# Existing endpoints

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

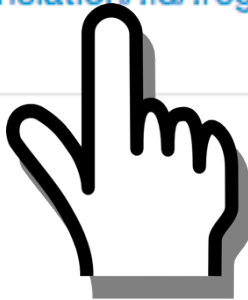
# Example

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
<a href="#">GET map/cdna/:id/:region</a>	Convert from cDNA coordinates to genomic coordinate API.
<a href="#">GET map/cds/:id/:region</a>	Convert from CDS coordinates to genomic coordinates
<a href="#">GET map/:species/:asm_one/:region/:asm_two</a>	Convert the co-ordinates of one <b>assembly</b> to another
<a href="#">GET map/translation/:id/:region</a>	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
<a href="#">GET overlap/id/:id</a>	Retrieves features (e.g. genes, transcripts, variations et
<a href="#">GET overlap/<b>region</b>/:species/:<b>region</b></a>	Retrieves multiple types of features for a given <b>region</b> .
<a href="#">GET overlap/translation/:id</a>	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](#)

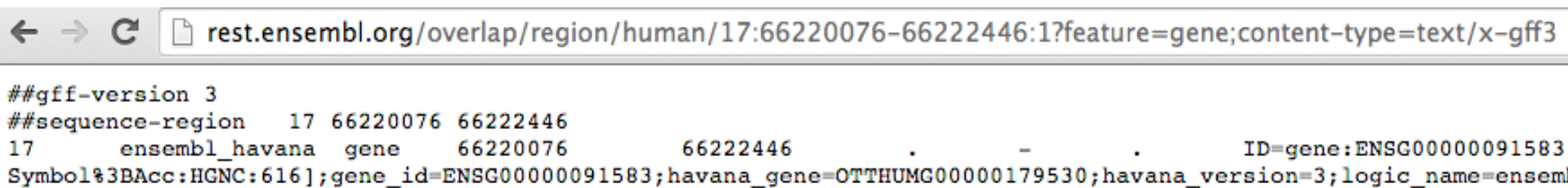
[Java](#)

[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 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 <a href="#">genetree/id/:id</a>	Retrieves a gene tree dump for a gene tree stable identifier
GET <a href="#">genetree/member/id/:id</a>	Retrieves a gene tree that contains the stable identifier
GET <a href="#">genetree/member/symbol/:species/:symbol</a>	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</confid
      <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
<a href="#">GET vep/:species/hgvs/:hgvs_notation</a>	Fetch <b>variant</b> consequences based on a HGVS notation
<a href="#">GET vep/:species/id/:id</a>	Fetch <b>variation</b> consequences based on a <b>variation</b> identifier
<a href="#">POST vep/:species/id</a>	Fetch <b>variation</b> consequences for multiple ids
<a href="#">GET vep/:species/region/:region/:allele/</a>	Fetch <b>variation</b> consequences
<a href="#">POST vep/:species/region</a>	Fetch <b>variation</b> consequences for multiple regions

## Variation



Resource	Description
<a href="#">GET variation/:species/</a>	Uses a <b>variation</b> identifier (e.g. rsID) to return the <b>variation</b> features
<a href="#">POST variation/:species/</a>	Uses a list of <b>variation</b> identifiers (e.g. rsID) to return the <b>variation</b> 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](#)

# 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](mailto: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](mailto:helpdesk@ensembl.org)

# Feedback

<http://tinyurl.com/Ensembl-REST>

# Ensembl Acknowledgements

## The Entire Ensembl Team

Fiona Cunningham<sup>1</sup>, M. Ridwan Amode<sup>1</sup>, Daniel Barrell<sup>1,2</sup>, Kathryn Beal<sup>1</sup>, Konstantinos Billis<sup>1</sup>, Simon Brent<sup>2</sup>, Denise Carvalho-Silva<sup>1</sup>, Peter Clapham<sup>2</sup>, Guy Coates<sup>2</sup>, Stephen Fitzgerald<sup>1</sup>, Laurent Gil<sup>1</sup>, Carlos García Girón<sup>1</sup>, Leo Gordon<sup>1</sup>, Thibaut Hourlier<sup>1</sup>, Sarah E. Hunt<sup>1</sup>, Sophie H. Janacek<sup>1</sup>, Nathan Johnson<sup>1</sup>, Thomas Juettemann<sup>1</sup>, Andreas K. Kähäri<sup>2</sup>, Stephen Keenan<sup>1</sup>, Fergal J. Martin<sup>1</sup>, Thomas Maurel<sup>1</sup>, William McLaren<sup>1</sup>, Daniel N. Murphy<sup>1,2</sup>, Rishi Nag<sup>1</sup>, Bert Overduin<sup>1</sup>, Anne Parker<sup>1</sup>, Mateus Patricio<sup>1</sup>, Emily Perry<sup>1</sup>, Miguel Pignatelli<sup>1</sup>, Harpreet Singh Riat<sup>1</sup>, Daniel Sheppard<sup>1</sup>, Kieron Taylor<sup>1</sup>, Anja Thormann<sup>1</sup>, Alessandro Vullo<sup>1</sup>, Steven P. Wilder<sup>1</sup>, Amonida Zadissa<sup>1</sup>, Bronwen L. Aken<sup>1</sup>, Ewan Birney<sup>1</sup>, Jennifer Harrow<sup>2</sup>, Rhoda Kinsella<sup>1</sup>, Matthieu Muffato<sup>1</sup>, Magali Ruffier<sup>1</sup>, Stephen M.J. Searle<sup>2</sup>, Giulietta Spudich<sup>1</sup>, Stephen J. Trevanion<sup>1</sup>, Andy Yates<sup>1</sup>, Daniel R. Zerbino<sup>1</sup> and Paul Flicek<sup>1,2,\*</sup>

<sup>1</sup>European Molecular Biology Laboratory, European Bioinformatics Institute, Wellcome Trust Genome Campus, Hinxton, Cambridge CB10 1SD, UK and <sup>2</sup>Wellcome 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

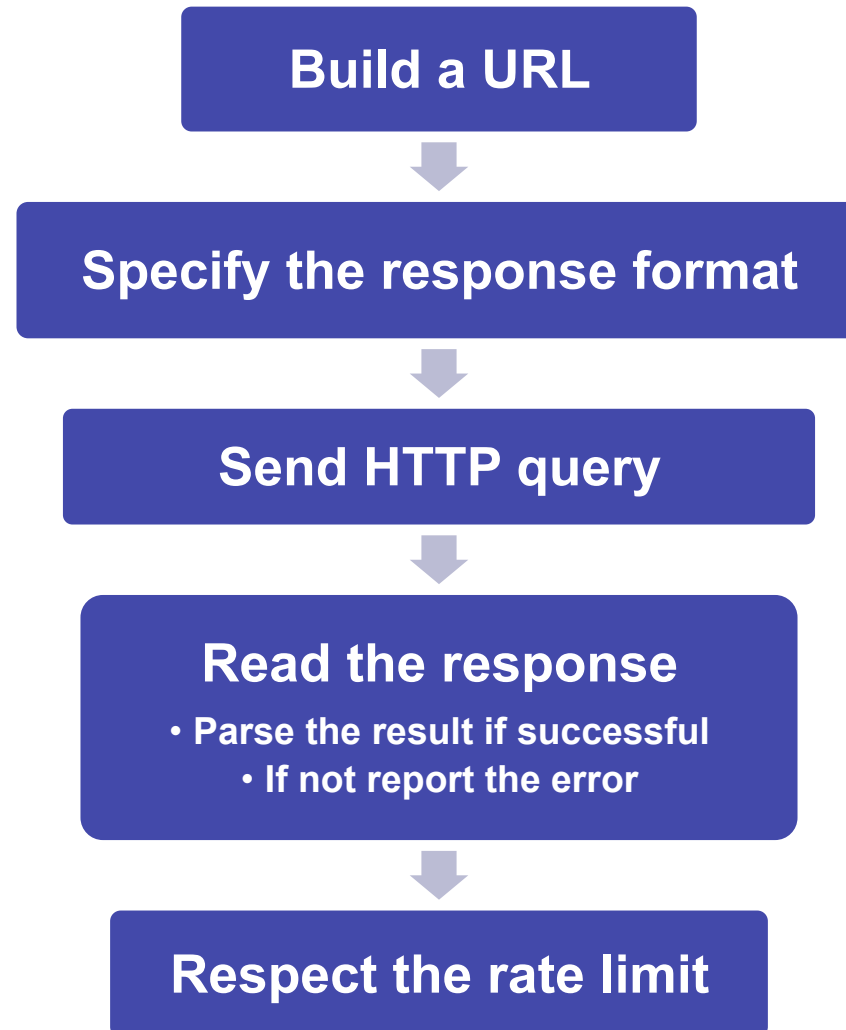
Species Required

# 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 3<sup>rd</sup> 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
<b>Endpoint</b>	A URL which will respond to a HTTP request and return data	/sequence/id/ ENST00000288602.fasta
<b>HTTP Header</b>	Sent to the server along side the requested URL.	Accepts: application/json Content-type: application/json
<b>HTTP Parameter</b>	Key value pairs separated by an = sign given to an endpoint after all required parameters.	?type=cds
<b>HTTP Method</b>	Tells the server the kind of operation you want to perform. Are you retrieving data or sending data to the server	GET POST
<b>HTTP Status Code</b>	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
<b>JSON</b>	Common cross-language data structure interchange format	{ “key”: [1, “two”]}

# Output Formats

Format	Content-Type	Extension	Notes
<b>FASTA</b>	text/x-fasta	.fasta	
<b>GFF3</b>	text/x-gff3	.gff3	
<b>JSON</b>	application/json	.json	Standard serialisation format
<b>JSONP</b>	text/javascript	.jsonp	Used to avoid browser sandbox issues. Use CORS instead
<b>Newick</b>	text/x-nh	.nh	Old tree format. Use PhyloXML instead
<b>SeqXML</b>	text/x-seqxml+xml	.seqxml	FASTA replacement
<b>PhyloXML</b>	text/x-phyloxml+xml	.phyloxml	Phylogenetic format
<b>Text</b>	text/plain	.txt	
<b>XML</b>	text/xml	.xml	
<b>YAML</b>	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
<b>Content-type</b>	MIME type	Describes what format the response is. The formats are the same as those used in the content-type request header
<b>Retry-After</b>	Floating point seconds	If found you must wait for this long before retrying the server
<b>X-RateLimit-Limit</b>	Integer	What the total limit of requests is
<b>X-RateLimit-Remaining</b>	Integer	How many requests you have left
<b>X-RateLimit-Reset</b>	Floating point seconds	How long before your tokens reset to the amount given in X-RateLimit-Limit
<b>X-Runtime</b>	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](mailto: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](mailto:helpdesk@ensembl.org)