Using Web services

Disclaimer about downloading genomic or protein sequences

If you need the genomic, transcript, protein sequences for many IDs (genes, transcripts, genomic sequence/chromosomes), instead of using webservices please check the ready files at each website, eg: https://plasmodb.org/common/downloads/Current_Release.

E.g. https://plasmodb.org/common/downloads/Current_Release/Pfalciparum3D7/fasta/data/PlasmoDB-65_Pfalciparum3D7_AnnotatedProteins.fasta

Record types and searches

Our websites data are organized around :

1. A small set of **record types** (e.g. genes, transcripts, datasets, organisms, etc).
   a. eg: https://plasmodb.org/plasmo/service/record-types
2. A record type defines:
   a. **attributes** (univalue)
   b. **tables** (multivalue)
   c. **reporters** which allow for different output formats in a search result
   d. **searches**
   e. eg: https://plasmodb.org/plasmo/service/record-types/genomic-sequence
3. There is a containment relationship between some record types:
   a. **Organism > Genomic-Sequences > Genes > Transcripts**.
4. The **searches** for each record type:
   a. define input **parameters**, whose values are needed to run.
   b. return **IDs** (`primary_key`) of a record type, and for each ID you may request any of the **attributes** and **tables**.
   c. sometimes they define "**dynamic attributes**" whose value for each ID is calculate on the fly. These attributes are not in the record definition; they show in the search definition.
   d. eg: https://plasmodb.org/plasmo/service/record-types/transcript/searches/GenesByGoTerm (find the dynamic attribute "evidence_code")
5. **Reporters** to choose from**,** are:
   a. `JSON` : you may request any number of attributes and tables
   b. `AttributeTabular` : tab delimited; any number of attributes
   c. `TableTabular` : tab delimited; one table
   d. `FASTA`
   e. `GFF3`

API

The full API of WDK REST service is at: https://veupathdb.org/service-api.html .

Below are the basic ones you need, to get search results the same way you run searches in the website and obtain results.

Record types definitions

https://plasmodb.org/plasmo/service/record-types

https://plasmodb.org/plasmo/service/record-types/gene

Be aware that searches may define ID attributes calculated on the fly. These would be described under the search definition. These are not included in the ontology.

Searches definitions

https://plasmodb.org/plasmo/service/record-types/transcript/searches

https://plasmodb.org/plasmo/service/record-types/transcript/searches/GeneByLocusTag

Ontology of records and searches

A file contains the ontology we use to categorize searches, records, attributes and tables. It lists all searches, attributes and tables used by the WDK client and sets scopes for the website client to determine where to show what.

https://github.com/VEuPathDB/ApiCommonModel/blob/master/Model/lib/wdk/ontology/individuals.txt

A subset is uploaded to: Sep 2023 - individuals project_home - genes/transcripts/organisms

It is useful to know these if you use the website to check on record types and their attributes and tables. But take into account the ontology is not an exhaustive list of defined searches, records, attributes or tables.

Genes and Transcripts searches

Genes and Transcripts are closely related. In the website we offer “gene searches” that return transcripts records; they are really transcript searches under the hood. But when using web services you need to understand the difference.

1. In **gene search** results: genes `primary_key` is composed of 2 strings:
   a. `source_id` , `project_id` . E.g. PF3D7_1133400, PlasmoDB;
   b. `source_id` is the gene ID;
   c. `project_id` is the component website that contains these data.
2. In **transcript search** results: transcripts `primary_key` is composed of 3 strings:
   a. `gene_source_id` , `source_id` , `project_id` . E.g. PF3D7_1133400, PF3D7_1133400.1, PlasmoDB;
   b. `source_id` is the transcript ID.
3. The **transcript record** defines most attributes, while the **gene record** defines all tables and some attributes. The attributes offered by genes and transcripts records may overlap.
4. To get just **one table for all IDs in a result**, you may use a **transcript search**: the transcript record offers a `TableTabular` reporter which will convert to genes and will allow you to download a gene table, one at a time.
5. **Use a gene search returning genes** if you want any number of **both attributes and tables in a result** (using the json reporter). There are only 2 gene searches:
   a. `single_record_question_GeneRecordClasses_GeneRecordClass` : takes a gene ID and a project ID as input
   b. `GenesByTaxonGene` : takes organisms as input

Check each search help page described in next section to find the correct syntax in your GET or POST. Eg: : https://plasmodb.org/plasmo/app/web-services-help?searchName=single_record_question_GeneRecordClasses_GeneRecordClass&weight=10&primaryKeys=PF3D7_1133400%2CPlasmoDB .

Web service request syntax for a search: GET and POST

In a website, each search page (e.g. https://plasmodb.org/plasmo/app/search/transcript/GenesByTaxon) has a link (under the “Get Answer” button) to a web services help page to build **a GET and a POST**. You can use either one. This is explained in this help page: https://plasmodb.org/plasmo/app/static-content/content/PlasmoDB/webServices.html.

For example, the search https://plasmodb.org/plasmo/app/search/transcript/GenesByTaxon, after selecting an organism (eg: *Plasmodium falciparum* 3D7) the link takes you to https://plasmodb.org/plasmo/app/web-services-help?searchName=GenesByTaxon&weight=10&organism=%5B%22Plasmodium%20falciparum%203D7%22%5D where you can define your output and copy the proposed GET or POST.

Next we show you some GET examples using different reporters (standard == json).

Get all organisms in a site, and for each: the URLs to the organism fasta files, its component website and its gene count

https://plasmodb.org/plasmo/service/record-types/organism/searches/GenomeDataTypes/reports/standard?reportConfig={"attributes":["primary_key","species","URLGenomeFasta","URLproteinFasta","project_id","genecount"],"tables":[],"attributeFormat":"text"}

For each organism, get all the gene IDs and transcript IDs with the specified attributes

https://plasmodb.org/plasmo/service/record-types/transcript/searches/GenesByTaxon/reports/standard?organism=["Plasmodium falciparum 3D7"]&reportConfig={"attributes":["primary_key","organism","gene_product","transcript_product"],"tables":[],"attributeFormat":"text"}

For each transcript ID, get the specified attributes

https://plasmodb.org/plasmo/service/record-types/transcript/searches/single_record_question_TranscriptRecordClasses_TranscriptRecordClass/reports/attributesTabular?primaryKeys=PF3D7_1133400%2CPF3D7_1133400.1%2CPlasmoDB&reportConfig={"attributes":["primary_key","source_id","gene_transcript_count","strand","gene_type","transcript_length","genomic_sequence_length","chromosome","sequence_id","organism","gene_ortholog_number","gene_paralog_number","protein_length","cds_length","interpro_id","pfam_id","pirsf_id","smart_id","ec_numbers","predicted_go_id_function","annotated_go_id_function"],"includeHeader":true,"attachmentType":"plain"}

For each gene ID, get the specified tables

https://plasmodb.org/plasmo/service/record-types/gene/searches/single_record_question_GeneRecordClasses_GeneRecordClass/reports/standard?primaryKeys=PF3D7_1133400%2CPlasmoDB&reportConfig={"attributes":["primary_key"],"tables":["GeneTranscripts","GeneModelDump","PreferredProducts","GeneLinkouts","GeneLocation","OrthologsLite","InterPro","GOTerms"],"attributeFormat":"text"}

For each organism, get all genes with the specified tables

https://plasmodb.org/plasmo/service/record-types/gene/searches/GenesByTaxonGene/reports/standard?organism=["Plasmodium falciparum 3D7"]&reportConfig={"attributes":["primary_key","organism","product"],"tables":["GeneTranscripts","GeneModelDump","PreferredProducts","GeneLinkouts","GeneLocation","OrthologsLite","InterPro","GOTerms"],"attributeFormat":"text"}

Using the portal *vs* a component site

There are 12 component sites and one portal: VEuPathDB.org. The component sites are: AmoebaDB, CryptoDB, FungiDB, GiardiaDB, HostDB, MicrosporidiaDB, PiroplasmaDB, PlasmoDB, ToxoDB, TrichDB, TriTrypDB and VectorBase. For most record types (all but dataset and organism), when running a search, the portal reaches out to component sites to get the search results. That means **it will be faster to use a component site directly when you can**. For a given ID (a gene, a sequence, a pathway), the portal has access to most attributes (univalue) information locally but does not have access to tables (multivalue) information; **if you need table information for an ID, use a**

component site. If you have an ID in your hand and you want to know to which component site it belongs, you may query the portal and get back a `project_id` as part of the IDs in the result; this will tell you which component site to query next.

The best use of the portal is to get a table with **all organisms in our sites, and for each organism: the component site, and the urls to access their fasta and gff files**.

https://veupathdb.org/veupathdb/service/record-types/organism/searches/GenomeDataTypes/reports/standard?reportConfig={"attributes":["primary_key","species","URLGenomeFasta","URLproteinFasta","project_id","genecount"],"tables": [],"attributeFormat":"text"}

Appendices

Url encoding

HTML URL Encoding Reference

Gene and Transcript basic attributes and tables

```
GENE ATTR

source_id
name

organism:       italics: P. falciparum 3D7
organism_text
organism_full

chromosome
sequence_id
strand_plus_minus
start_min
end_max
location_text       Pf3D7_11_v3:1,292,966..1,296,696(+)

transcript_count
exon_count

gene_type, type_with_pseudo
product

ortholog_number
paralog_number

gene_gff_link

TRANSCR ATTR

source_id
gene_source_id

organism        italics: Plasmodium falciparum 3D7
organism_full

chromosome
sequence_id
strand
gene_start_min
```

```
gene_end_max
gene_location_text, location_text Pf3D7_11_v3:1,292,966..1,296,696(+) (transcript no commas)

genomic_sequence_length
cds_length
five_prime_utr_length
transcript_length
protein_length

gene_transcript_count
exon_count
gene_exon_count

gene_type
gene_product
transcript_product

gene_ortholog_number
gene_paralog_number

interpro_description
interpro_id
pfam_description
pfam_id
smart_description
smart_id
uniprot_ids
uniprot_links

annotated_go_function
predicted_go_function
ec_numbers

transcript_sequence
protein_sequence


GENE TABLES

GeneId
GeneName

GeneTranscripts

GeneLocation
GeneModelDump

PreferredProducts
Product (dumper?)
AlternateProducts

GeneLinkouts

EcNumber
ECNumbers
GOTerms
GeneGffGoTerms

```

```
InterPro

Orthologs
OrthologsLite

ProteinProperties

ProteinSequence

Sequences

```

Example of a single search POST request and response

************** SINGLE RECORD QUESTION REQUEST **************

POST https://veupathdb.org/veupathdb/service/record-types/dataset/searches/single_record_question_DatasetRecordClasses_DatasetRecordClass/reports/standard

```
{
  "searchConfig": {
    "parameters": {
      "primaryKeys": "DS_09e81bd198"
    }
  },
  "reportConfig": {
    "attributes": [
      "version","contact","institution","description","eupath_release",
      "acknowledgement","annotation_version","newcategory","caveat",
      "email","functional_annotation_version","genome_version",
      "megabase_pairs","organism_prefix","release_policy","summary","genecount"
    ],
    "tables": [
      "Publications","Contacts","GenomeHistory","DatasetHistory","Version",
      "References","HyperLinks","GeneTypeCounts","TranscriptTypeCounts"
    ]
  }
}
```

************** SINGLE RECORD QUESTION RESPONSE **************

```
{
  "records": [{
    "id": [{
      "name": "dataset_id",
      "value": "DS_09e81bd198"
    }],
    "displayName": "EC mappings",
    "recordClassName": "DatasetRecordClasses.DatasetRecordClass",
    "attributes": {
      "megabase_pairs": null,
      "summary": "Enzyme classification (EC) mappings to <i>P. falciparum</i> genes supplied by Hagai Ginsburg (
      "acknowledgement": null,
      "annotation_version": null,
      "genome_version": null,
      "description": "Enzyme classification (EC) mappings to <i>P. falciparum</i> genes supplied by Hagai Ginsbu
      "organism_prefix": "<i>Plasmodium falciparum</i> 3D7<br>",
      "version": "03-2019",
      "release_policy": null,
      "newcategory": "Function prediction",
      "institution": "Malaria Parasite Metabolic Pathways",
      "contact": "Hagai Ginsburg",
      "genecount": null,
      "eupath_release": "EuPathDB rel. 1.0, 01-JAN-05",
      "caveat": null,
      "email": "hagai@vms.huji.ac.il",
      "functional_annotation_version": null
    },
    "tables": {
      "GenomeHistory": [],
      "Publications": [1],
      "Contacts": [1],
      "Version": [1],
      "GeneTypeCounts": [],
      "References": [3],
      "DatasetHistory": [],
      "HyperLinks": [],
      "TranscriptTypeCounts": []
    },
    "tableErrors": []
  }],
  "meta": {
    "responseCount": 1,
    "totalCount": 1,
    "viewTotalCount": 1,
    "displayTotalCount": 1,
    "displayViewTotalCount": 1,
    "cachePreviouslyExisted": false,
    "sorting": [],
    "recordClassName": "dataset",
    "pagination": {
      "offset": 0,
      "numRecords": -1
    },
    "attributes": [
      "version",
      "contact",
      "institution",
      "description",
      "eupath_release",
      "acknowledgement",
      "annotation_version",
      "newcategory",
      "caveat",
      "email",
      "functional_annotation_version",
      "genome_version",
      "megabase_pairs",
      "organism_prefix",
      "release_policy",
      "summary",
      "genecount"
    ],
    "tables": [
      "Publications",
      "Contacts",
      "GenomeHistory",
      "DatasetHistory",
      "Version",
      "References",
      "HyperLinks",
      "GeneTypeCounts",
      "TranscriptTypeCounts"
    ]
  }
}
```

Example of a single record POST

************** RECORD'S ENDPOINT REQUEST **************

POST https://veupathdb.org/veupathdb/service/record-types/dataset/records

```
{
  "primaryKey":[{
    "name":"dataset_id",
    "value":"DS_09e81bd198"
  }],
  "attributes":[
    "version","contact","institution","description","eupath_release",
    "acknowledgement","annotation_version","newcategory","caveat",
    "email","functional_annotation_version","genome_version",
    "megabase_pairs","organism_prefix","release_policy","summary","genecount"
  ],
  "tables":[
    "Publications","Contacts","GenomeHistory","DatasetHistory","Version",
    "References","HyperLinks","GeneTypeCounts","TranscriptTypeCounts"
  ]
}
```

************** RECORD'S ENDPOINT RESPONSE **************

```
{
  "id": [{
    "name": "dataset_id",
    "value": "DS_09e81bd198"
  }],
  "displayName": "EC mappings",
  "recordClassName": "DatasetRecordClasses.DatasetRecordClass",
  "attributes": {
    "megabase_pairs": null,
    "summary": "Enzyme classification (EC) mappings to <i>P. falciparum</i> genes supplied by Hagai Ginsburg (MP
    "acknowledgement": null,
    "annotation_version": null,
    "genome_version": null,
    "description": "Enzyme classification (EC) mappings to <i>P. falciparum</i> genes supplied by Hagai Ginsburg
    "organism_prefix": "<i>Plasmodium falciparum</i> 3D7<br>",
    "version": "03-2019",
    "release_policy": null,
    "newcategory": "Function prediction",
    "institution": "Malaria Parasite Metabolic Pathways",
    "contact": "Hagai Ginsburg",
    "genecount": null,
    "eupath_release": "EuPathDB rel. 1.0, 01-JAN-05",
    "caveat": null,
    "email": "hagai@vms.huji.ac.il",
    "functional_annotation_version": null
  },
  "tables": {
    "GenomeHistory": [0],
    "Publications": [1],
    "Contacts": [1],
    "Version": [1],
    "GeneTypeCounts": [],
    "References": [3],
    "DatasetHistory": [],
    "HyperLinks": [],
    "TranscriptTypeCounts": []
  },
  "tableErrors": []
}
```

wget, curl examples

Curl and wget are both command-line tools used to retrieve data from internet. They use different protocols to perform this task, with curl supporting a variety of protocols, including HTTP, HTTPS, FTP, FTPS, SCP, SFTP, and more. Wget, on other hand, primarily supports HTTP and FTP protocols, and in general more limited in options, but faster.

```
wget --debug --header 'content-type: application/json' --post-data '{"searchConfig": {"parameters": {"organism":
```

```
curl -d '{"attributes": [],"primaryKey": [{"name": "source_id","value": "ACON2_034749"},{"name": "project_id","va
```