Collaboratory

grEPI

Global Repository of Epidemiological Parameters

API documentation

May 2026, v3

1. Overview

grEPI provides a public REST API that returns data in JSON format. The API is read-only (GET requests only) and does not require authentication or a Collaboratory account. There are four endpoints, each corresponding to a core data domain in grEPI.

Endpoint	Description
`EpiParameterEstimates`	Epidemiological parameter estimates extracted from the literature (returns the latest version of each estimate only)
`EpiParameterEstimates/{grEPI_ID}/Diff`	Differences between two versions of an epidemiological parameter estimate
`PublicArticle`	Literature records (articles, preprints) from which parameter estimates have been extracted
`PathogenTaxonomy`	Pathogen taxonomy reference data with grEPI preferred names and ICD classification codes
`DiseaseCausedbyPathogen`	Disease-to-pathogen mappings with grEPI preferred names and ICD classification codes

The interactive API documentation (Swagger UI) is available at:
https://collaboratory.who.int/grepi/api/swagger/index.html

The underlying OpenAPI specification (JSON) is available at:
https://collaboratory.who.int/grepi/api/swagger/v1/swagger.json

How requests work

All requests are HTTP GET, meaning the API is read-only and used solely for retrieving data. The base URL for all requests is:

```
https://collaboratory.who.int/grepi/api
```

Append the endpoint name (e.g. `EpiParameterEstimates`, `PublicArticle`) and any query parameters to construct a request. For example:

```
https://collaboratory.who.int/grepi/api/PublicArticle?Article_Publication_Year=2024
&Author=Miura
```

All query parameters are optional. When no parameters are provided, the endpoint returns all records. When multiple parameters are provided, they are combined using AND logic, so only records matching all specified criteria are returned.

Response format

Paginated endpoints (all except the `EpiParameterEstimates/{grEPI_ID}/Diff` endpoint) support two response shapes, depending on whether pagination is requested.

Bare array (no pagination). If `page` and `pageSize` are both omitted, or if only one of them is supplied, the endpoint returns a flat JSON array of records. All fields for every record are returned on every call.

Envelope (pagination). If both `page` and `pageSize` are supplied, the endpoint returns a JSON object with the following shape:

```
{
  "data": [ ... ],
  "total": 192,
  "page": 1,
  "pageSize": 50,
  "totalPages": 4
}
```

- `data` is an array of record objects. All fields for every record are returned on every call.
- `total` is the total number of records matching the query, across all pages.
- `page` is the page number returned.
- `pageSize` is the requested page size.
- `totalPages` is the total number of pages available for the query.

If `page` exceeds `totalPages`, `data` will be empty.

Pagination is controlled by two query parameters: `page` (page number, integer) and `pageSize` (records per page, integer). Here is an example of a paginated request:

```
https://collaboratory.who.int/grepi/api/EpiParameterEstimates?Disease_Name_Preferred=Measles&page=2&pageSize=50
```

Use query parameters to filter where possible to limit the size of the result set.

The `EpiParameterEstimates/{grEPI_ID}/Diff` endpoint returns a single object describing the comparison between two versions. See Section 4.5.

String matching

String filters are case-insensitive and support partial matching. For example, all of the following return results for Measles:

```
Disease_Name_Preferred=Measles

Disease_Name_Preferred=measles

Disease_Name_Preferred=measle
```

Date format

Date parameters (such as `Created_Date_From` and `Last_Modified_Date_From`) accept dates in ISO 8601 format: `YYYY-MM-DDTHH:MM:SS`. For example: `2026-01-01T00:00:00`.

2. Getting Started

This section walks through common tasks using the grEPI API from R and Python. No API key or authentication is needed. All examples can be run as-is.

R (using httr2)

Your first API call

Install `{httr2}` if you haven't already, then retrieve all measles parameter estimates:

```
# install.packages("httr2")
library(httr2)


resp <- request("https://collaboratory.who.int/grepi/api/EpiParameterEstimates") |>
  req_url_query(
    Disease_Name_Preferred = "Measles”
  ) |>
  req_perform()


body <- resp_body_json(resp, simplifyVector = TRUE)
measles <- body


# Check what came back
cat("Fields:", ncol(measles), "\n")
```

Selecting columns

The API returns all fields. To work with a subset, select columns after retrieval:

```
measles[, c("grEPI_ID", "epi_Parameter_Category_Name",
            "epiParameter_Estimate_Type",
            "epiParameter_Estimate_Subtype",
            "epiParameter_Estimate_Point_Value",
            "epiParameter_Estimate_Unit")]
```

Combining filters

Retrieve only Case Fatality Ratio estimates for measles:

```
resp <- request("https://collaboratory.who.int/grepi/api/EpiParameterEstimates") |>
  req_url_query(
    Disease_Name_Preferred = "Measles",
    Estimate_Type_Name = "Case Fatality Ratio (CFR)"
```

```
  ) |>
  req_perform()


cfr <- resp_body_json(resp, simplifyVector = TRUE)
```

Searching the literature

Search articles by title keyword and publication year. The `Article_Title` filter supports partial, case-insensitive matching:

```
resp <- request("https://collaboratory.who.int/grepi/api/PublicArticle") |>
  req_url_query(Article_Title = "mpox", Article_Publication_Year = 2024) |>
  req_perform()


articles <- resp_body_json(resp, simplifyVector = TRUE)
```

Looking up an article by DOI

DOIs should be provided without the https://doi.org/ prefix:

```
resp <- request("https://collaboratory.who.int/grepi/api/PublicArticle") |>
  req_url_query(Article_DOI = "10.1016/j.vaccine.2022.02.057") |>
  req_perform()


article <- resp_body_json(resp, simplifyVector = TRUE)
```

Using reference data endpoints

Retrieve the full disease-pathogen mapping and pathogen taxonomy:

```
# Disease-pathogen mapping
resp <- request("https://collaboratory.who.int/grepi/api/DiseaseCausedbyPathogen")
|>
  req_perform()
mapping <- resp_body_json(resp, simplifyVector = TRUE)


# Pathogen taxonomy for a specific family
resp <- request("https://collaboratory.who.int/grepi/api/PathogenTaxonomy") |>
  req_url_query(Pathogen_Family_Name_grEPI_preferred = "Paramyxovirus") |>
  req_perform()
taxonomy <- resp_body_json(resp, simplifyVector = TRUE)
```

Discovering valid values

To find which diseases, estimate types, or article labels are available:

```
# Valid disease names
```

```
resp <- request("https://collaboratory.who.int/grepi/api/DiseaseCausedbyPathogen")
|>
  req_perform()
diseases <- resp_body_json(resp, simplifyVector = TRUE)
unique(diseases$disease_Name_grEPI_preferred)


# Valid estimate types for a specific disease
resp <- request("https://collaboratory.who.int/grepi/api/EpiParameterEstimates") |>
  req_url_query(Disease_Name_Preferred = "Mpox") |>
  req_perform()
mpox <- resp_body_json(resp, simplifyVector = TRUE)
unique(mpox$epiParameter_Estimate_Type)
```

Working with nested fields

Some fields contain nested data. `epi_Parameter_Population_Country_list` is an array of objects (each with `country_Name`, `country_Code_ISO2`, `country_Code_ISO3`, `country_World_Continent`, `country_WHO_Region`). `article_Authors` is a flat array of name strings. `epi_Parameter_Population_Sex_List` and `epi_Parameter_Pathogen_Transmissions_List` are also arrays.

With `{jsonlite}`'s `simplifyVector = TRUE`, nested arrays become list columns. To access them:

```
# Country data for the first record
measles$epi_Parameter_Population_Country_list[[1]]
```

Python (using requests and pandas)

Your first API call

Install `requests` and `pandas` if you haven't already, then retrieve all measles parameter estimates:

```
import requests
import pandas as pd

url = "https://collaboratory.who.int/grepi/api/EpiParameterEstimates"
params = {
    "Disease_Name_Preferred": "Measles"
}
response = requests.get(url, params=params)
response.raise_for_status()

body = response.json()
measles = pd.DataFrame(body)

print(f"Fields: {len(measles.columns)}")
```

Selecting columns

```
measles[["grEPI_ID", "epi_Parameter_Category_Name",
         "epiParameter_Estimate_Type",
         "epiParameter_Estimate_Point_Value",
         "epiParameter_Estimate_Unit"]].head()
```

Combining filters

```
params = {
    "Disease_Name_Preferred": "Measles",
    "Estimate_Type_Name": "Case Fatality Ratio (CFR)"
}
response = requests.get(url, params=params)
cfr = pd.DataFrame(response.json())
```

Searching the literature

```
url = "https://collaboratory.who.int/grepi/api/PublicArticle"
params = {"Article_Title": "mpox", "Article_Publication_Year": 2024}
response = requests.get(url, params=params)
articles = pd.DataFrame(response.json())
```

Looking up an article by DOI

```
# Use the DOI without the https://doi.org/ prefix

params = {"Article_DOI": "10.1093/infdis/jiad091"}

response = requests.get("https://collaboratory.who.int/grepi/api/PublicArticle",
params=params)

article = pd.DataFrame(response.json())
```

Using reference data endpoints

```
# Full disease-pathogen mapping

response =
requests.get("https://collaboratory.who.int/grepi/api/DiseaseCausedbyPathogen")

mapping = pd.DataFrame(response.json())


# Valid disease names

mapping["disease_Name_grEPI_preferred"].unique()
```

Working with nested fields

```
# Nested fields example: population country data

response = requests.get(

    "https://collaboratory.who.int/grepi/api/EpiParameterEstimates",

    params={"Disease_Name_Preferred": "Measles"}

)


# Access country data for the first record

response.json()[0]["epi_Parameter_Population_Country_list"]


# To flatten country data into separate rows:

data = response.json()

data_with_countries = [

    r for r in data

    if r.get("epi_Parameter_Population_Country_list")

]



countries = pd.json_normalize(

    data_with_countries,

    record_path=["epi_Parameter_Population_Country_list"],

    meta=["grEPI_ID"],
```

```
    errors="ignore"
)
```

3. Using grEPI Data with epiparameter

The `{epiparameter}` R package (developed by the Epiverse-TRACE initiative) provides a standardized class for working with epidemiological parameters across Epiverse analysis packages and pipelines. Direct integration between grEPI and `{epiparameter}` is planned for the future. In the meantime, you can manually construct `<epiparameter>` objects from grEPI API data.

For full documentation on `{epiparameter}`, see: https://epiverse-trace.github.io/epiparameter/

Step 1: Retrieve a parameter estimate from grEPI

Start by pulling a specific estimate from the API. This example retrieves incubation period estimates for mpox:

```
library(httr2)
library(epiparameter)

# Retrieve incubation period estimates for mpox
resp <- request("https://collaboratory.who.int/grepi/api/EpiParameterEstimates") |>
  req_url_query(
    Disease_Name_Preferred = "Mpox",
    Estimate_Subtype_Name = "Incubation period"
  ) |>
  req_perform()

inc_data <- resp_body_json(resp, simplifyVector = TRUE)

# Look at what is available
inc_data[, c("grEPI_ID", "article_Label",
             "epiParameter_Estimate_Point_Value",
             "epiParameter_Estimate_Unit",
             "epiParameter_Distribution_Type")]

# Select an incubation period estimate record
rec <- inc_data[inc_data$article_Label == "Kroger 2023" &
                inc_data$epiParameter_Distribution_Type == "Log-normal", ][1, ]
```

Step 2: Construct an `<epiparameter>` object

Map grEPI fields to the `epiparameter()` constructor. The level of detail you include depends on what the grEPI record contains. At minimum, you need a disease name, an epidemiological parameter name, and either distribution parameters or summary statistics.

```
# grEPI distribution names use different conventions to R.
# This lookup maps grEPI names to the names accepted by {epiparameter}.
grepi_dist_lookup <- c(
  "Log-normal"  = "lnorm",
  "Gamma"       = "gamma",
  "Weibull"     = "weibull"
)

# Build the <epiparameter> object from grEPI fields.
# This record has a directly reported mean and standard deviation,
# so {epiparameter} can calculate the distribution parameters
# (meanlog/sdlog) automatically.
grepi_eparam <- epiparameter(
  disease = as.character(rec$disease_Name_Preferred),
  pathogen = as.character(rec$pathogen_Species_Name_Preferred),
  epi_name = "incubation_period",
  prob_distribution =
unname(grepi_dist_lookup[rec$epiParameter_Distribution_Type]),
  summary_stats = create_summary_stats(
    mean = as.numeric(rec$epiParameter_Estimate_Point_Value),
    sd = as.numeric(rec$epiParameter_Uncertainty_Single_Value)
  ),
  citation = create_citation(
    author = as.character(rec$article_Authors[[1]]),
    year = as.integer(rec$article_Publication_Year),
    title = as.character(rec$article_Title),
    journal = as.character(rec$literature_Source_Name),
    doi = as.character(rec$article_DOI)
  ),
  metadata = create_metadata(
```

```
    units = tolower(gsub(" .*", "", rec$epiParameter_Estimate_Unit)),
    sample_size = as.integer(rec$epi_Parameter_Population_Sample_Size)
  ),
  notes = paste("grEPI ID:", rec$grEPI_ID)
)


# Inspect the result
grepi_eparam
```

Note: Not all grEPI records contain the same summary statistics. The example above works because the selected record has a directly reported mean and standard deviation. Other records may have different combinations (e.g. mean with a credible interval, or median with a range). Before attempting conversion, check what the record contains in the `epiParameter_Estimate_Value_Type`, `epiParameter_Uncertainty_Value_Type`, and `epiParameter_Uncertainty_Single_Value` fields. Records without sufficient summary statistics for `{epiparameter}` to calculate distribution parameters can still be stored as unparameterised `<epiparameter>` objects by omitting the `prob_distribution` argument. Refer to `{epiparameter}` documentation for further information: https://epiverse-trace.github.io/epiparameter/

Step 3: Use in an analysis pipeline

Once you have an `<epiparameter>` object, it works with the standard `{epiparameter}` functions and can be passed to other Epiverse-TRACE packages. For example, you can generate random draws from the distribution, calculate density values, or use it as an input to transmission models:

```
# Generate random draws from the distribution
generate(grepi_eparam, times = 10)


# Calculate density, CDF, and quantiles
density(grepi_eparam, at = 5)
cdf(grepi_eparam, q = 7)
quantile(grepi_eparam, p = c(0.025, 0.5, 0.975))


# Plot the distribution
plot(grepi_eparam)
```

Converting multiple records

To convert a batch of grEPI records, wrap the construction logic in a function and apply it across rows. This is useful when you want to compare multiple estimates for the same parameter:

```
grepi_to_epiparameter <- function(rec, dist_lookup) {
  # Only convert records with a point estimate and supported distribution
  if (is.na(rec$epiParameter_Estimate_Point_Value)) return(NULL)

  # Map distribution name if available
  r_dist <- if (!is.na(rec$epiParameter_Distribution_Type) &&
                rec$epiParameter_Distribution_Type %in% names(dist_lookup)) {
    unname(dist_lookup[rec$epiParameter_Distribution_Type])
  } else {
    NA_character_
  }

  # Determine SD: use directly reported value if available,
  # otherwise skip
  sd_val <- if (!is.na(rec$epiParameter_Uncertainty_Value_Type) &&
                rec$epiParameter_Uncertainty_Value_Type == "SD (Standard
Deviation)" &&
                !is.na(rec$epiParameter_Uncertainty_Single_Value)) {
    as.numeric(rec$epiParameter_Uncertainty_Single_Value)
  } else {
    NA_real_
  }

  tryCatch(
    epiparameter(
      disease = as.character(rec$disease_Name_Preferred),
      pathogen = as.character(rec$pathogen_Species_Name_Preferred),
      epi_name = "incubation_period",
      prob_distribution = r_dist,
      summary_stats = create_summary_stats(
        mean = if (rec$epiParameter_Estimate_Value_Type == "Mean")
                 as.numeric(rec$epiParameter_Estimate_Point_Value) else NA_real_,
        median = if (rec$epiParameter_Estimate_Value_Type == "Median")
```

```
                    as.numeric(rec$epiParameter_Estimate_Point_Value) else NA_real_,
        sd = sd_val
      ),
      citation = create_citation(
        author = as.character(rec$article_Authors[[1]]),
        year = as.integer(rec$article_Publication_Year),
        title = as.character(rec$article_Title),
        journal = as.character(rec$literature_Source_Name),
        doi = as.character(rec$article_DOI)
      ),
      metadata = create_metadata(
        units = tolower(gsub(" .*", "", rec$epiParameter_Estimate_Unit)),
        sample_size = as.integer(rec$epi_Parameter_Population_Sample_Size)
      )
    ),
    error = function(e) {
      message("Skipping record ", rec$grEPI_ID, ": ", e$message)
      NULL
    }
  )
}

# Apply across all incubation period records
eparam_list <- lapply(seq_len(nrow(inc_data)), function(i) {
  grepi_to_epiparameter(inc_data[i, ], grepi_dist_lookup)
})

# Remove NULLs (records that could not be converted)
eparam_list <- Filter(Negate(is.null), eparam_list)
cat("Successfully converted", length(eparam_list), "of", nrow(inc_data),
"records\n")
```

4. Endpoint Reference

This section provides the full technical detail for each endpoint: query parameters, response field names, and data types. Fields are grouped by their naming prefix for readability.

4.1 EpiParameterEstimates

GET `https://collaboratory.who.int/grepi/api/EpiParameterEstimates`

This endpoint returns only the latest version of each parameter estimate. To retrieve differences between two versions of a specific estimate, use the `EpiParameterEstimates/{grEPI_ID}/Diff` endpoint (Section 4.5).

Query parameters

Parameter	Type	Description
`Disease_Name_Preferred`	string	Filter by disease name (e.g. "Measles", "Mpox")
`Pathogen_Species_Name_Preferred`	string	Filter by pathogen species (e.g. "Measles virus")
`Pathogen_Strain_Preferred`	string	Filter by pathogen strain
`Pathogen_Sub_Strain_Preferred`	string	Filter by pathogen sub-strain
`Estimate_Type_Name`	string	Filter by estimate type (e.g. "Case Fatality Ratio (CFR)")
`Estimate_Subtype_Name`	string	Filter by estimate subtype
`Epi_Delay_Time_From`	string	Filter by event type for the start of the interval (e.g., "Exposure", "Infection", "Symptom Onset")
`Epi_Delay_Time_To`	string	Filter by event type for the end of the interval (e.g., "Symptom Onset", "Reporting", "Recovery")
`Created_Date_From`	datetime	Records created on or after this date (YYYY-MM-DDTHH:MM:SS)
`Created_Date_To`	datetime	Records created on or before this date
`page`	integer	Page number to retrieve.
`pageSize`	integer	Number of records per page.

Response fields

Over 140 fields are returned per record. The main field groups, by naming prefix, are:

Identification: `grEPI_ID` (UUID).

disease_* fields: `disease_Name_Preferred, disease_Code_ICD, disease_Sub_Chapter_Name_ICD, disease_Chapter_Name_ICD, disease_Chapter_Code_ICD, disease_Spread_Type`.

pathogen_* fields: `pathogen_Species_Name_Preferred, pathogen_Species_Code_ICD, pathogen_Genus_Name_Preferred, pathogen_Family_Name_Preferred`.

epiParameter_Estimate_* fields: `epiParameter_Estimate_Type, epiParameter_Estimate_Subtype, epiParameter_Estimate_Value_Type` (e.g. "Median", "Crude proportion"), `epiParameter_Estimate_Point_Value` (numeric), `epiParameter_Estimate_Range_Lower_Bound_Value, epiParameter_Estimate_Range_Upper_Bound_Value, epiParameter_Estimate_Unit`. For delay parameters: `epiParameter_EventFrom` and `epiParameter_EventTo`.

epi_Parameter_Category_Name: the parameter category (e.g. "Severity", "Delays").

epiParameter_Uncertainty_* fields: `epiParameter_Uncertainty_Value_Type, epiParameter_Uncertainty_Single_Value, epiParameter_Uncertainty_Range_Lower_Bound_Value, epiParameter_Uncertainty_Range_Upper_Bound_Value, epiParameter_Uncertainty_Unit`.

epiParameter_Distribution_* fields: `epiParameter_Distribution_Type`. Up to three distribution parameters, each with value, value type, and uncertainty fields, following the pattern `epiParameter_Distribution_Parameter1_Value` through `epiParameter_Distribution_Parameter3_Value`.

epi_Parameter_Method_* fields: `epi_Parameter_Method_PercentageUnit_Calculation` (e.g. "Naive"), `epi_Parameter_Method_Inference`, and related boolean fields for censoring, truncation, bias adjustment, and discretization.

epi_Parameter_PercentageUnit_* fields: for proportion-based estimates: `epi_Parameter_PercentageUnit_numerator, epi_Parameter_PercentageUnit_denominator`.

epi_Parameter_CaseClassification_* fields: `epi_Parameter_CaseClassification_isConfirmed, epi_Parameter_CaseClassification_isProbable, epi_Parameter_CaseClassification_isSuspected, epi_Parameter_CaseClassification_isUnspecified` (all boolean).

epi_Parameter_Population_* fields: `epi_Parameter_Population_Sample_Size` (integer), age fields (min/max in years and months), `epi_Parameter_Population_Sex_List` (nested array), `epi_Parameter_Population_Setting, epi_Parameter_Population_Group, epi_Parameter_Population_Comment` (free text). Study period: `epi_Parameter_Population_Study_Start_Year / epi_Parameter_Population_Study_End_Year`. `epi_Parameter_Population_Country_list` is a nested array of objects, each containing `country_Name, country_Code_ISO2, country_Code_ISO3, country_World_Continent, country_WHO_Region`.

article_* fields: `article_Title, article_DOI` (e.g. "10.1016/j.vaccine.2022.02.057"), `article_Label` (short citation key, e.g. "Bose 2022"), `article_Publication_Year` (integer), `literature_Source_Name`, `article_Authors` (flat array of name strings).

epi_Parameter_DataSource_* fields:
`epi_Parameter_Data_ExtractedBy_OrganizationGroup_Name, epi_Parameter_DataSource_Primary_ImportedFrom_Name, epi_Parameter_DataSource_Primary_ImportedFrom_Project, epi_Parameter_DataSource_Name.`

Versioning fields: Each record represents the latest version of the parameter estimate and includes four fields describing the version state:

`epi_Parameter_Record_Version_ID` (UUID): unique identifier for this version of the record.
`epi_Parameter_Record_Version_Latest_Number` (integer): the version number (1 for the original, 2 for the first revision, and so on).
`epi_Parameter_Record_Version_Latest_Comment` (string, nullable): an optional note describing what changed in this version.
`epi_Parameter_Record_Version_Latest_CreatedDate` (datetime): when this version was created.

To retrieve differences between two versions of an estimate, use the `EpiParameterEstimates/{grEPI_ID}/Diff` endpoint (Section 4.5).

4.2 PublicArticle

GET `https://collaboratory.who.int/grepi/api/PublicArticle`

Returns article records from the grEPI literature database.

Query parameters

Parameter	Type	Description
`Article_DOI`	string	Filter by DOI (e.g. "10.1093/infdis/jiad091")
`Article_Title`	string	Filter by article title (partial, case-insensitive match)
`Article_Label`	string	Filter by short citation key (e.g. "Miura 2024")
`Article_Publication_Year`	integer	Filter by publication year (e.g. 2024)
`Author`	string	Filter by author name
`Literature_Source_Name`	string	Filter by journal or source name (e.g. "Lancet")
`Literature_Source_Type_Name`	string	Filter by source type (e.g. "Journal")
`page`	integer	Page number to retrieve.
`pageSize`	integer	Number of records per page.

Response fields

`article_ID` (UUID), `article_DOI`, `article_Title`, `article_Label`, `article_Publication_Year` (integer), `article_Paper_Version_Only` (boolean), `article_Page_First` / `article_Page_Last` (integer), `literature_Source_Name`, `literature_Source_Type_Name`, `article_grEPI_RecordStatus`, `article_grEPI_LastModifiedDate` (datetime).

`article_Authors` is a nested array of objects containing: `article_Author_ID` (UUID), `person_ID` (UUID), `person_First_Names`, `person_Last_Names`, `authorOrder` (integer). Note: on the `EpiParameterEstimates` endpoint, `article_Authors` is returned as a flat array of name strings rather than structured objects.

4.3 PathogenTaxonomy

GET `https://collaboratory.who.int/grepi/api/PathogenTaxonomy`

Returns pathogen taxonomy records with both grEPI preferred names and ICD classification codes.

Query parameters

Parameter	Type	Description
`Pathogen_Type_Name_ICD`	string	Filter by pathogen type (ICD name, e.g. "Virus")
`Pathogen_Family_Name_grEPI_preferred`	string	Filter by family (e.g. "Poxvirus")
`Pathogen_Genus_Name_grEPI_preferred`	string	Filter by genus (e.g. "Orthopoxvirus")
`Pathogen_Species_Name_grEPI_preferred`	string	Filter by species (e.g. "Monkeypox virus")
`Pathogen_Strain_Name_grEPI_preferred`	string	Filter by strain
`Pathogen_SubStrain_Name_grEPI_preferred`	string	Filter by sub-strain
`Last_Modified_Date_From`	datetime	Records modified on or after this date (YYYY-MM-DDTHH:MM:SS)
`Last_Modified_Date_To`	datetime	Records modified on or before this date
`page`	integer	Page number to retrieve.
`pageSize`	integer	Number of records per page.

Response fields

`pathogen_Species_ID` (UUID). For each taxonomic level (Type, Family, Genus, Species, Strain, SubStrain), four fields are returned: `pathogen_[Level]_Name_grEPI_preferred` (the grEPI name), `pathogen_[Level]_Name_ICD` (the ICD name), `pathogen_[Level]_Code_ICD` (the ICD code), and `pathogen_[Level]_Identifier_ICD` (the ICD identifier). For example: `pathogen_Family_Name_grEPI_preferred, pathogen_Family_Name_ICD, pathogen_Family_Code_ICD, pathogen_Family_Identifier_ICD`. Also includes `diseaseCausedByPathogen_grEPI_RecordStatus` and `diseaseCausedByPathogen_grEPI_LastModifiedDate`.

4.4 DiseaseCausedbyPathogen

GET `https://collaboratory.who.int/grepi/api/DiseaseCausedbyPathogen`

Returns the full mapping between diseases and their causative pathogens.

Query parameters

Parameter	Type	Description
`Last_Modified_Date_From`	datetime	Records modified on or after this date (YYYY-MM-DDTHH:MM:SS)
`Last_Modified_Date_To`	datetime	Records modified on or before this date
`page`	integer	Page number to retrieve.
`pageSize`	integer	Number of records per page.

Response fields

`disease_Pathogen_ID` (UUID). Disease fields: `disease_Name_grEPI_preferred`, `disease_Name_ICD`, `disease_Code_ICD`, `disease_Identifier_ICD`, plus sub-chapter and chapter fields. Pathogen fields follow the same four-field-per-level pattern described for `PathogenTaxonomy` above. Also includes `diseaseCausedByPathogen_grEPI_RecordStatus` and `diseaseCausedByPathogen_grEPI_LastModifiedDate`.

4.5 EpiParameterEstimates Diff

GET `https://collaboratory.who.int/grepi/api/EpiParameterEstimates/{grEPI_ID}/Diff`

Returns the differences between two versions of a single epidemiological parameter estimate. This is the API equivalent of the “Compare versions” view in the grEPI web application. Pagination is not applied to this endpoint, as each request returns the diff for a single record.

Path parameters

Parameter	Type	Description
`grEPI_ID`	string (UUID)	Identifier of the parameter estimate to compare versions for

Query parameters

Parameter	Type	Description
`Epi_Parameter_Version_Number`	integer	First version number in the comparison (e.g., 1)
`Epi_Parameter_Version_Number_ComparedTo`	integer	Second version number to compare against (e.g., 4)

Response fields

The `Diff` endpoint returns a single JSON object with the following structure:

```
{
  "grEPI_ID": "63201b76-f1b5-476e-af58-6167c7f6f954",
  "epi_Parameter_Version_Number": 1,
  "epi_Parameter_Version_Number_ComparedTo": 3,
  "changes": [
    {
      "field": "Estimate Category Comment",
      "epi_Parameter_Version_Number_value": "-",
      "epi_Parameter_Version_Number_ComparedTo_value": "v2 adding"
    },
    {
      "field": "Uncertainty Range Upper",
      "epi_Parameter_Version_Number_value": "10.9",
      "epi_Parameter_Version_Number_ComparedTo_value": "-"
    }
  ]
}
```

Top-level fields

Field	Type	Description
`grEPI_ID`	string (UUID)	Identifier of the parameter estimate
`epi_Parameter_Version_Number`	integer	The first version in the comparison (as supplied in the request)
`epi_Parameter_Version_Number_ComparedTo`	integer	The second version in the comparison (as supplied in the request)
`changes`	array	One object per field that differs between the two versions. An empty array means the two versions are identical.

Fields inside each `changes` entry

Field	Type	Description
`field`	string	Human-readable label for the field that differs (e.g., "Inference Method", "Study Start")
`epi_Parameter_Version_Number_value`	string	The value of this field in the first version. A dash ("-") indicates the field had no value in that version.
`epi_Parameter_Version_Number_ComparedTo_value`	string	The value of this field in the second version. A dash ("-") indicates the field had no value in that version.

Values are returned as strings, including for numeric and date fields. Where a field was added in the second version, the first version's value will be "-"; where a field was removed, the second version's value will be "-".