Onyx

CLIMB-TRE

None

Table of contents

1. Onyx - API for Pathogen Metadata

1.1 Introduction

This is the documentation for Onyx, a database and API for managing sample metadata, their analyses, and other associated data.

As part of CLIMB-TRE, Onyx serves as the central metadata repository for the following projects:

- mSCAPE (Metagenomics Surveillance Collaboration and Analysis Programme)
  A collaborative initiative led by UKHSA, involving a consortium of NHS and academic partners, to deliver a pilot surveillance network trialling the use of metagenomic data for public health surveillance and pathogen analysis.
- PATH-SAFE (Pathogen Surveillance in Agriculture, Food and Environment)
  Led by the FSA, PATH-SAFE piloted the development of a national surveillance network to improve the detection and tracking of foodborne human pathogens and AMR within agriculture.
- synthSCAPE (Synthetic dataset for mSCAPE)
- openMGS (Open Meta-Genomic Surveillance)

1.2 Contents

2. Types & Lookups

2.1 Types

Types in Onyx define the various categories of data which can be stored.

Each field belongs to a certain type. This dictates what kind of data the field can store (e.g. text, numbers, dates, etc.), as well as what filter operations (i.e. lookups) can be carried out on values of the field.

2.1.1 `text`

`[exact]` `[ne]` `[in]` `[notin]` `[contains]` `[startswith]` `[endswith]` `[iexact]` `[icontains]` `[istartswith]` `[iendswith]` `[length]` `[length__in]` `[length__range]` `[isnull]`

A string of characters.

Examples: `"C-1234567890"`, `"Details about something"`

2.1.2 `choice`

`[exact]` `[ne]` `[in]` `[notin]` `[isnull]`

A restricted set of options.

Examples: `"ENG"`, `"WALES"`, `"SCOT"`, `"NI"`

2.1.3 `integer`

`[exact]` `[ne]` `[in]` `[notin]` `[lt]` `[lte]` `[gt]` `[gte]` `[range]` `[isnull]`

A whole number.

Examples: `1`, `-1`, `123`

2.1.4 `decimal`

`[exact]` `[ne]` `[in]` `[notin]` `[lt]` `[lte]` `[gt]` `[gte]` `[range]` `[isnull]`

A decimal number.

Examples: `1.234`, `1.0`, `23.456`

2.1.5 `date`

`[exact]` `[ne]` `[in]` `[notin]` `[lt]` `[lte]` `[gt]` `[gte]` `[range]` `[iso_year]` `[iso_year__in]` `[iso_year__range]` `[week]` `[week__in]` `[week__range]` `[isnull]`

A date.

Examples: `"2023-03"`, `"2023-04-05"`, `"2024-01-01"`

2.1.6 `datetime`

`[exact]` `[ne]` `[in]` `[notin]` `[lt]` `[lte]` `[gt]` `[gte]` `[range]` `[iso_year]` `[iso_year__in]` `[iso_year__range]` `[week]` `[week__in]` `[week__range]` `[isnull]`

A date and time.

Examples: `"2023-01-01 15:30:03"`, `"2024-01-01 09:30:17"`

2.1.7 `bool`

`[exact]` `[ne]` `[in]` `[notin]` `[isnull]`

A true or false value.

Examples: `True`, `False`

2.1.8 `relation`

`[isnull]`

A link to a row, or multiple rows, in another table.

2.1.9 `array`

`[exact]` `[contains]` `[contained_by]` `[overlap]` `[length]` `[length__in]` `[length__range]` `[isnull]`

A list of values.

Examples: `[1, 2, 3]`, `["hello", "world", "!"]`

2.1.10 `structure`

`[exact]` `[contains]` `[contained_by]` `[has_key]` `[has_keys]` `[has_any_keys]` `[isnull]`

An arbitrary JSON structure.

Examples: `{"hello" : "world", "goodbye" : "!"}`, `{"numbers" : [1, 2, {"more_numbers" : [3, 4, 5]}]}`

2.2 Lookups

Lookups can be used to specify more complex conditions that fields must match when filtering.

Different types have different lookups available to them.

2.2.1 `exact`

`[text]` `[choice]` `[integer]` `[decimal]` `[date]` `[datetime]` `[bool]` `[array]` `[structure]`

Return values equal to the search value.

2.2.2 `ne`

`[text]` `[choice]` `[integer]` `[decimal]` `[date]` `[datetime]` `[bool]`

Return values not equal to the search value.

2.2.3 `in`

`[text]` `[choice]` `[integer]` `[decimal]` `[date]` `[datetime]` `[bool]`

Return values that are within the set of search values.

2.2.4 `notin`

`[text]` `[choice]` `[integer]` `[decimal]` `[date]` `[datetime]` `[bool]`

Return values that are not within the set of search values.

2.2.5 `contains`

`[text]` `[array]` `[structure]`

Return values that contain the search value.

2.2.6 `startswith`

`[text]`

Return values that start with the search value.

2.2.7 `endswith`

`[text]`

Return values that end with the search value.

2.2.8 `iexact`

`[text]`

Return values case-insensitively equal to the search value.

2.2.9 `icontains`

`[text]`

Return values that case-insensitively contain the search value.

2.2.10 `istartswith`

`[text]`

Return values that case-insensitively start with the search value.

2.2.11 `iendswith`

`[text]`

Return values that case-insensitively end with the search value.

2.2.12 `length`

`[text]` `[array]`

Return values with a length equal to the search value.

2.2.13 `length__in`

`[text]` `[array]`

Return values with a length that is within the set of search values.

2.2.14 `length__range`

`[text]` `[array]`

Return values with a length that is within an inclusive range of search values.

2.2.15 `lt`

`[integer]` `[decimal]` `[date]` `[datetime]`

Return values less than the search value.

2.2.16 `lte`

`[integer]` `[decimal]` `[date]` `[datetime]`

Return values less than or equal to the search value.

2.2.17 `gt`

`[integer]` `[decimal]` `[date]` `[datetime]`

Return values greater than the search value.

2.2.18 `gte`

`[integer]` `[decimal]` `[date]` `[datetime]`

Return values greater than or equal to the search value.

2.2.19 `range`

`[integer]` `[decimal]` `[date]` `[datetime]`

Return values within an inclusive range of search values.

2.2.20 `iso_year`

`[date]` `[datetime]`

Return values with an ISO 8601 week-numbering year equal to the search year.

2.2.21 `iso_year__in`

`[date]` `[datetime]`

Return values with an ISO 8601 week-numbering year that is within the set of search years.

2.2.22 `iso_year__range`

`[date]` `[datetime]`

Return values with an ISO 8601 week-numbering year that is within an inclusive range of search years.

2.2.23 `week`

`[date]` `[datetime]`

Return values with an ISO 8601 week number equal to the search number.

2.2.24 `week__in`

`[date]` `[datetime]`

Return values with an ISO 8601 week number that is within the set of search numbers.

2.2.25 `week__range`

`[date]` `[datetime]`

Return values with an ISO 8601 week number that is within an inclusive range of search numbers.

2.2.26 `isnull`

`[text]` `[choice]` `[integer]` `[decimal]` `[date]` `[datetime]` `[bool]` `[relation]` `[array]` `[structure]`

Return values that are empty (`isnull = True`) or non-empty (`isnull = False`).

- For `text` and `choice` types, 'empty' is defined as the empty string `""`.
- For the `relation` type, 'empty' is defined as there being zero items linked to the record being evaluated.
- For the `array` type, 'empty' is defined as the empty array `[]`.
- For the `structure` type, 'empty' is defined as the empty structure `{}`.
- For all other types, 'empty' is the SQL `null` value.

2.2.27 `contained_by`

`[array]` `[structure]`

Return values that are equal to, or a subset of, the search value.

2.2.28 `overlap`

`[array]`

Return values that overlap with the search value.

2.2.29 `has_key`

`[structure]`

Return values that have a top-level key which contains the search value.

2.2.30 `has_keys`

`[structure]`

Return values that have top-level keys which contains all of the search values.

2.2.31 `has_any_keys`

`[structure]`

Return values that have top-level keys which contains any of the search values.