CLIMB-TRE

CLIMB-TRE

None

Table of contents

1. CLIMB-TRE

The CLIMB Trusted Research Environment (CLIMB-TRE) project provides tools with which users can upload metagenomics data, with metadata, and analyse them on CLIMB.

This site documents how to use the CLIMB-TRE tools and is distinct from more general documentation on using CLIMB. Read further for general information on how to upload or analyse data.

Last modified 2024-03-25 15:29:15+00:00 (ac20a21)

2. Getting started

2.1 Uploading data

2.1.1 Overview

Data in CLIMB-TRE is managed through a database called Onyx. To upload data into Onyx, you must deposit the appropriate files (including the metadata) into the relevant S3 bucket on CLIMB. We recommend doing this using the AWS or `s3cmd` command-line tools. For general information about how to upload data to CLIMB, see the CLIMB docs on setting up `s3cmd` locally and running `s3cmd` locally or on Bryn. You may also wish to review the overall CLIMB storage documentation.

Each CLIMB-TRE project requires data (e.g. FASTQ sequencing reads) and metadata (e.g. a CSV file). These must match the relevant specification ("spec") and be uploaded to the appropriate S3 bucket. Doing so will trigger the ingest process. Data that doesn't meet the spec will not be ingested.

Lines starting with `$` indicate commands to be entered at a terminal. The `$` represents the prompt, which might be different on your system.

2.1.2 Preparing example FASTQ files

As an example, let's imagine we want to upload the two example files in Conor Meehan's Pathogen genomics course as part of the mSCAPE project. The two files are from Hikichi et al. (2019), `DRR187559_1.fastqsanger.bz2` and `DRR187559_2.fastqsanger.bz2`, available in this Zenodo archive. You can download the files either by clicking on them in the Zenodo interface or with the common command line tools `wget`:

```
$ wget https://zenodo.org/record/4534098/files/DRR187559_1.fastqsanger.bz2
$ wget https://zenodo.org/record/4534098/files/DRR187559_2.fastqsanger.bz2
```

or `curl`:

```
$ curl -L https://zenodo.org/record/4534098/files/DRR187559_1.fastqsanger.bz2 -O
$ curl -L https://zenodo.org/record/4534098/files/DRR187559_2.fastqsanger.bz2 -O
```

These two files are bzip2 files, not gzip, which is what we need. We can convert them by piping the output from `bzcat` (which decompresses the files) to `gzip -c` (which compresses the stream and writes it to `STDOUT`) and then to new files:

```
$ bzcat DRR187559_1.fastqsanger.bz2 | gzip -c > DRR187559_1.fastq.gz
$ bzcat DRR187559_2.fastqsanger.bz2 | gzip -c > DRR187559_2.fastq.gz
```

The mSCAPE specification says that our files must have names like `mscape.[run_index].[run_id].[extension]`, where the extension is `1.fastq.gz` or `2.fastq.gz`. The `run_index` and `run_id` can in principle contain any alphanumeric characters, underscores (`_`) or hyphens ('-'), so you can rename the FASTQ files to whatever meets those requirements. At the command line, this means moving the files with something like:

```
$ mv DRR187559_1.fastq.gz mscape.test-run-index-01.test-run-id-01.1.fastq.gz
$ mv DRR187559_2.fastq.gz mscape.test-run-index-01.test-run-id-01.2.fastq.gz
```

2.1.3 Creating a metadata CSV file

Data uploads require that the FASTQ files are accompanied by a CSV file with the metadata (e.g. when the sample was taken, what type of sample it is). This CSV file must have two rows:

1. the headers, as in the project metadata specification; and
2. the actual metadata.

It's filename must match the FASTQ files but with the extension `csv` instead of `1.fastq.gz` or `2.fastq.gz` (or `fastq.gz` if you're trying a single read upload.

For the sake of our test and getting to know the system, you should try to create such a file by hand by referring to the relevant metadata spec. The columns are documented in alphabetical order but can be given in any order. The optional columns can be omitted entirely.

Note that the `run_index` and `run_id` must *exactly* match the values implied by the FASTQ filenames. E.g., in my example above

- the `run_index` is `test-run-index-01` and
- the `run_id` is `test-run-id-01`.

The first few columns of your metadata CSV file might look like

```
run_index,run_id,biosample_id,sample_source,sample_type,...
test-run-index-01,test-run-id-01,test-sample-01,nose_and_throat,swab,...
```

with no extra spaces separating the fields.

2.1.4 Uploading files to S3 buckets

You're now ready to upload your data to one of the buckets, which we'll do using the `s3cmd` tool. There's more information about using `s3cmd` with Bryn in the CLIMB-BIG-DATA documentation on storage.

You can download `s3cmd` from the `s3cmd` download pages or install it using `pip` (perhaps in a virtual or Conda environment) with

```
$ python3 -m pip install s3cmd
```

To set `s3cmd` up to communicate with the buckets, you'll need your API keys from Bryn. You can find them by logging in to Bryn, selecting the S3 Buckets tab on the left and click the Show API Keys button that appears below the list of buckets.

You can then set up `s3cmd` with

```
$ s3cmd --configure
```

When asked for the following, you should give these answers:

- Access Key: value of `AWS_ACCESS_KEY_ID` displayed on Bryn.
- Secret Key: value of `AWS_SECRET_ACCESS_KEY` displayed on Bryn.
- Default Region [US]: leave blank.
- S3 Endpoint: `s3.climb.ac.uk`
- DNS-style bucket+hostname:port template for accessing a bucket: `%(bucket)s.s3.climb.ac.uk`

You now should be ready to upload the data. But where? The names of the S3 buckets for each project are given in the metadata specs but are usually of the form

```
[project]-[sequencing_org]-[platform]-[test_flag]
```

We'll use `mscape-public-illumina-test`, so the command to "put" the three files in the bucket would be

```
$ s3cmd put mscape.test-run-index-01.test-run-id-01.csv mscape.test-run-index-01.test-run-id-01.1.fastq.gz mscape.test-run-index-01.test-run-id-01.2.fastq.gz
s3://mscape-public-illumina-test
```

You should then see the progress of your upload (the files might be split into parts), after which you're back at the terminal.

Now what?

2.1.5 Finding the result of your upload

You won't get any feedback from `s3cmd` about the progress of your data into Onyx. When the data is received in the bucket, it announces the files to whoever is listening, which includes a program called Roz. It then starts to check the data: Are all the files present? Are they named correctly? Is the metadata well-formed? If so, the data is copied into internal project buckets and a record is added to the database, Onyx.

At this point, you can interact with your data through Onyx, which is described in the page on analysing data in Onyx.

Last modified 2024-03-25 15:29:15+00:00 (ac20a21)

2.2 Analysing data

2.2.1 Overview

Once data and metadata have been ingested into the Onyx database, you can query it using the Onyx client, which provides a command line interface (CLI) and Python API. This short example demonstrates a few principal functions. More are described in the `onyx-client` documentation.

This guide also assumes that you're using a Notebook Server on CLIMB, so that once installed, the Onyx client will automatically be configured.

2.2.2 Onyx client basics

First, let's install the Onyx client, which is available through the conda-forge package `climb-onyx-client` and can thus be installed with `conda`. As advised in the CLIMB docs on installing software, you should install the client in a new Conda environment. I'll name my environment `onyx` and install `climb-onyx-client`, as well as `ipykernel` (so that the client is available in my Jupyter Notebooks).

```
jovyan:~$ conda create -n onyx ipykernel climb-onyx-client
```

Let's activate this environment.

```
jovyan:~$ conda activate onyx
```

On Bryn's Notebook Servers, the client will automatically be configured. Try running the command-line client with

```
(onyx) jovyan:~$ onyx
```

This should show you some options and commands that are available. Have a look at your own profile with

```
(onyx) jovyan:~$ onyx profile
```

and which projects you have access to with

```
(onyx) jovyan:~$ onyx projects
```

You should see `mscape` listed.

2.2.3 Querying data

As an example task, we'll see if we can find any sequencing data performed for ZymoBIOMICS sources. These are designed with a particular specification of DNA from eight bacteria and two yeasts. We can use these to see if our protocol correctly recovers the DNA fractions. I.e. if our protocol is biased.

From the command line, the main route to querying Onyx is via the `filter` command. On its own, this queries the database with *no* filters. The command

```
(onyx) jovyan:~$ onyx filter mscape
```

will produce tens of thousands of lines of JSON, so let's not do that just yet. To first see which fields are available in the database, we can use

```
(onyx) jovyan:~$ onyx fields mscape
...
├────────────────────────────────┼──────────┼────────────────────┼───────────────────────────────────────────────────────────────────────────────┼────────────────
│ extraction_enrichment_protocol │ optional │ text               │ Details of nucleic acid extraction and optional enrichment steps.
│                                                                               │
├────────────────────────────────┼──────────┼────────────────────┼───────────────────────────────────────────────────────────────────────────────┼────────────────
...
```

Let's search the database for entries with `zymo` (case-insensitive) in this field.

```
(onyx) jovyan:~$ onyx filter mscape --field extraction_enrichment_protocol.icontains=zymo
...
```

That should return JSON data for a few entries. You may wish to format the data as CSV or TSV with `--format csv` or `--format tsv`, respectively.

2.2.4 Inspecting some pipeline output on the command line

When data is ingested into Onyx, a taxonomic classification is automatically run. The last part of the JSON data is usually some of this, in JSON format. The complete reports can be found in the S3 buckets given in the `'taxon_report'` field. You can find this in the output you've already produced or modify the `filter` command to only request them using the `--include` flag. e.g.

```
(onyx) jovyan:~$ onyx filter mscape --field extraction_enrichment_protocol.icontains=zymo --include=taxon_reports
[
    {
        "taxon_reports": "s3://mscape-published-taxon-reports/C-FDE50853AD/"
    },
    {
        "taxon_reports": "s3://mscape-published-taxon-reports/C-04F4495068/"
    }
]
```

Multiple fields can be requested with the `--include` flag e.g.

```
(onyx) jovyan:~$ onyx filter mscape --field extraction_enrichment_protocol.icontains=zymo --include climb_id,taxon_reports
[
    {
        "climb_id": "C-FDE50853AD",
        "taxon_reports": "s3://mscape-published-taxon-reports/C-FDE50853AD/"
    },
    {
        "climb_id": "C-04F4495068",
        "taxon_reports": "s3://mscape-published-taxon-reports/C-04F4495068/"
    }
]
```

You can conversely exclude individual fields using the `--exclude` flag in the same way.

Either way, you now have the location of the taxonomy reports. Let's have a look with `s3cmd`.

```
(onyx) jovyan:~$ s3cmd ls s3://mscape-published-taxon-reports/C-FDE50853AD/
2023-11-10 12:56   146K  s3://mscape-published-taxon-reports/C-FDE50853AD/PlusPF.kraken.json
2023-11-10 12:56     2G  s3://mscape-published-taxon-reports/C-FDE50853AD/PlusPF.kraken_assignments.tsv
2023-11-10 12:56   193K  s3://mscape-published-taxon-reports/C-FDE50853AD/PlusPF.kraken_report.txt
```

The plain text report is what we're after, so let's download that with `s3cmd`:

```
(onyx) jovyan:~$ s3cmd get s3://mscape-published-taxon-reports/C-FDE50853AD/PlusPF.kraken_report.txt
download: 's3://mscape-published-taxon-reports/C-FDE50853AD/PlusPF.kraken_report.txt' -> './PlusPF.kraken_report.txt'  [1 of 1]
 197750 of 197750   100% in    0s     3.79 MB/s  done
```

If you've never seen one of these reports before, it's worth having a quick look with a tool like `less` or by opening it using the JupyterLab file browser. For reference, it's worth showing the header

```
(onyx) jovyan:~$ head -n 1 PlusPF.kraken_report.txt
% of Seqs       Clades  Taxonomies      Rank    Taxonomy ID     Scientific Name
```

The Zymo sample is prepared with 12% *Bacillus subtilis*. Let's see how much was actually reported in the results:

```
(onyx) jovyan:~$ grep "Bacillus subtilis" PlusPF.kraken_report.txt
 20.30  435278  1452    G1      653685                  Bacillus subtilis group
  0.12  2624    1952    S       1423                      Bacillus subtilis
  0.03  565     242     S1      135461                      Bacillus subtilis subsp. subtilis
  0.01  108     108     S2      1404258                       Bacillus subtilis subsp. subtilis str. OH 131.1
  ...
```

Looks like 20.3%, though classified under *Bacillus subtilis* "subgroup", rather than *Bacillus subtilis*, which reportedly only comprises 0.12% of the sample. Most of that 20.3% is under *Bacillus spizizenii*.

An important detail here is that the fraction reported in this output is not calculated in the same way as what's used in the reference values (12% for bacteria; 2% for yeasts). Let's make a fairer comparison using the JSON taxonomic data.

2.2.5 Working with database output in Python

To fairly compare the taxonomic data with the reference values in the Zymo community, we need to know the proportions of gDNA, so we need to compute the number of base pairs that were assigned to each taxon. Let's make this comparison in Python using the Onyx client's Python API.

Let's first run the same query for the Zymo data. We'll follow the examples in the Onyx documentation and run the query in a context manager.

```
import os
from onyx import OnyxConfig, OnyxEnv, OnyxClient

config = OnyxConfig(
    domain=os.environ[OnyxEnv.DOMAIN],
    token=os.environ[OnyxEnv.TOKEN],
)

with OnyxClient(config) as client:
    records = list(client.filter(
        "mscape",
        fields={
            "extraction_enrichment_protocol__icontains": "zymo",
        },
    ))
```

We've wrapped the `filter` call in a `list` because otherwise we get a generator.

If you want to inspect the data, it's a bit easier to read if formatted with indentation, which can be done using the standard `json.dumps` function:

```
import json
print(json.dumps(records[0], indent=2))  # show first record
```

In each record, the `'taxa_files'` key gives us a list of dictionaries that each has a number of reads and a mean length, the product of which is the total number of base pairs that were read for that taxon. A simple first step is to convert the taxonomic data (for the first record) into a Pandas DataFrame with

```
import pandas as pd

df = pd.DataFrame(records[0]['taxa_files'])
```

We also need to drop a few lower-level taxa that are already accounted for in higher ones. e.g. the reads for *Bacillus spizizenii TU-B-10* are among the reads counted for *Bacillus spizizenii*. A quick way of doing this is by selecting the rows that have only two words in their names.

```
df = df.loc[df['human_readable'].apply(lambda name: len(name.split()) == 2)]
```

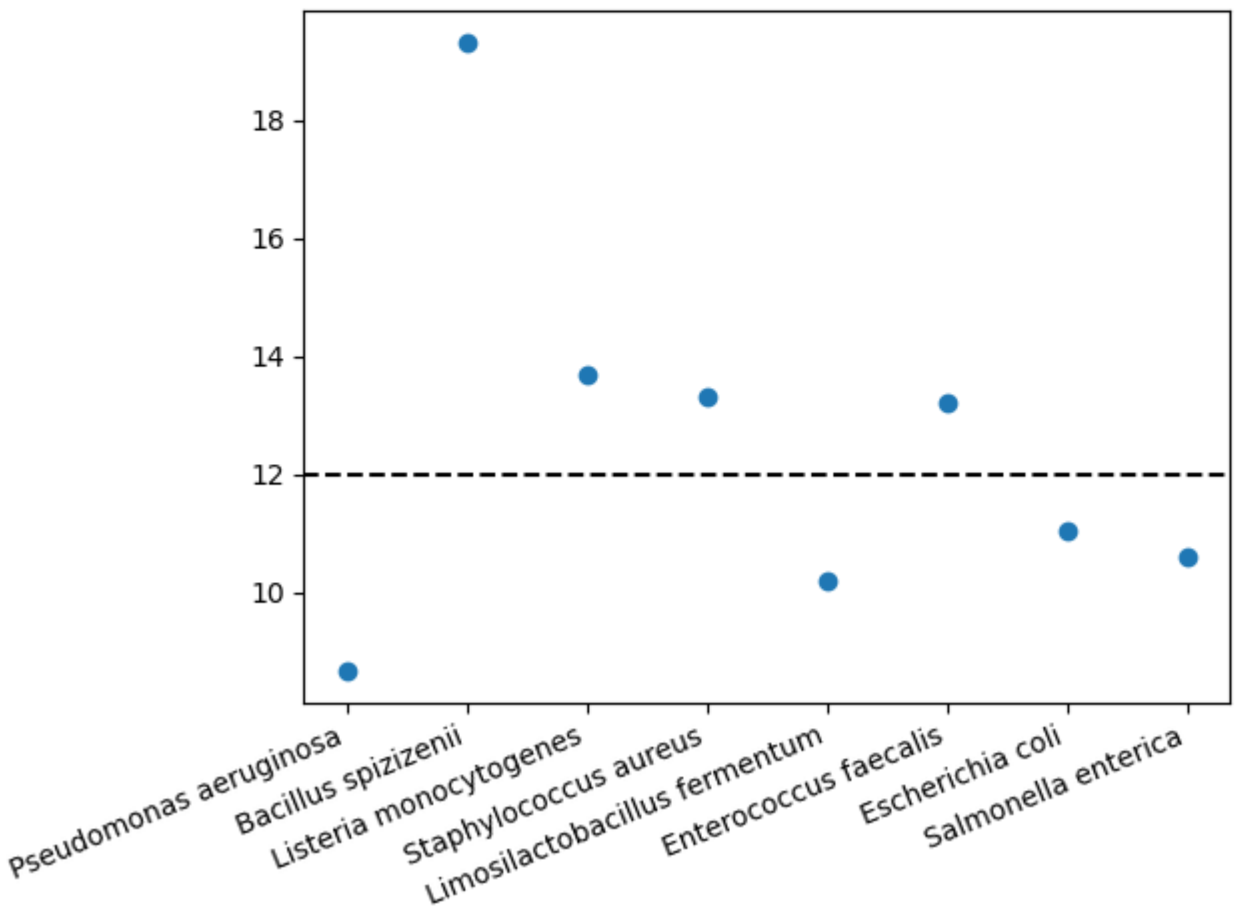
Now, let's add columns for the total number of base pairs associated with each taxon and what proportion that is of the total.

```
df['gDNA'] = df['n_reads']*df['mean_len']
df['proportion'] = df['gDNA']/df['gDNA'].sum()
```

Finally, let's make a rough plot with a black dashed line at 12%.

```
import matplotlib.pyplot as plt

plt.plot(df['human_readable'], df['proportion']*100, 'o')
plt.axhline(12, c='k', ls='--');
plt.xticks(rotation=22.5, ha='right');
```



There are some clear discrepancies—*Pseudomonas aeruginosa* is underreported and *Bacillus spizizenii* is overreported—but this matches results by e.g. Nicholls et al. (2019).

This short example is intended as a basic demonstration of what's possible in CLIMB-TRE. We're always interested to hear more examples of research questions that CLIMB-TRE can answer, so let us know if you have an example that could be included as a guide for others.

Last modified 2024-03-25 15:29:15+00:00 (ac20a21)

2.3 Further Examples

2.3.1 Analysis examples for mSCAPE

Retrieve all samples that contain a particular taxa e.g. `pseudomonas`

This can be done through the CLI:

```
$ onyx filter mscape --field taxa_files.human_readable.icontains=pseudomonas --format csv
```

Or through the Python API:

```python
import os
from onyx import OnyxConfig, OnyxClient, OnyxEnv, OnyxField

config = OnyxConfig(
    domain=os.environ[OnyxEnv.DOMAIN],
    token=os.environ[OnyxEnv.TOKEN],
)

with OnyxClient(config) as client:
    # Filter for read sets containing "pseudomonas"
    for metadata in client.query(
        "mscape",
        query=OnyxField(taxa_files__human_readable__icontains="pseudomonas"),
    ):
        # Do analysis here
        print("CLIMB ID:", metadata["climb_id"])
        print("Published date:", metadata["published_date"])

        # The query command by default does not return taxonomic information
        # To get this, we have to call the `get` method and retrieve the samples individually
        full_metadata = client.get("mscape", metadata["climb_id"])

        # Now we can inspect the taxonomic information for the readset
        print(
            "Number of binned reads:", len(full_metadata["taxa_files"])
        )  # etc. Do more analysis
        print("Pseudomonas taxa:")
        for taxa in full_metadata["taxa_files"]:
            if "pseudomonas" in taxa["human_readable"].lower():
                print("-", taxa["human_readable"])
```

Example output for this python script:

```
CLIMB ID: C-FE89BACF2D
Published date: 2024-02-28
Number of binned reads: 3
Pseudomonas taxa:
- Pseudomonas aeruginosa
- Pseudomonas aeruginosa PA7
CLIMB ID: C-470A57DCD0
Published date: 2024-02-28
Number of binned reads: 8
Pseudomonas taxa:
- Pseudomonas aeruginosa
- Pseudomonas aeruginosa PA7
CLIMB ID: C-FB67513BE0
Published date: 2024-02-28
Number of binned reads: 4
Pseudomonas taxa:
- Pseudomonas aeruginosa
CLIMB ID: C-E49EED98E4
Published date: 2024-02-28
Number of binned reads: 3
Pseudomonas taxa:
- Pseudomonas aeruginosa
- Pseudomonas aeruginosa PA7
...
```

Get a CSV distribution of all binned taxa present in the dataset

Through the CLI:

```
$ onyx filter mscape --summarise taxa_files.taxon_id,taxa_files.human_readable --format csv
```

Or through the Python API:

```
import os
from onyx import OnyxConfig, OnyxClient, OnyxEnv, OnyxField

config = OnyxConfig(
    domain=os.environ[OnyxEnv.DOMAIN],
    token=os.environ[OnyxEnv.TOKEN],
)

with OnyxClient(config) as client:
    for summary_data in client.query(
        "mscape",
        summarise=["taxa_files__taxon_id", "taxa_files__human_readable"],
    ):
        # Do analysis here
        print("Taxon ID:", summary_data["taxa_files__taxon_id"])
        print("Taxon name:", summary_data["taxa_files__human_readable"])
        print("Number of readsets present:", summary_data["count"])
```

Example output for this python script:

```
Taxon ID: 1304
Taxon name: Streptococcus salivarius
Number of readsets present: 22
Taxon ID: 1305
Taxon name: Streptococcus sanguinis
Number of readsets present: 9
Taxon ID: 1313
Taxon name: Streptococcus pneumoniae
Number of readsets present: 26
Taxon ID: 1318
Taxon name: Streptococcus parasanguinis
Number of readsets present: 42
Taxon ID: 1328
Taxon name: Streptococcus anginosus
Number of readsets present: 4
...
```

Last modified 2024-03-25 15:29:15+00:00 (ac20a21)

3. Project specifications

3.1 Project specification structure

3.1.1 Overview

All projects on CLIMB-TRE are specified in the same way.

3.1.2 Files to be provided

These are the files that must be uploaded (usually some sequencing reads and a metadata file). Submissions without the correct number of files provided will be considered incomplete and will not be processed.

3.1.3 File naming convention

This is the convention to which the provided file names must adhere.

Each of the files to be provided will use the same basename followed by specified extensions (e.g. for data versus metadata). The basename for each file is usually several fields separated by a fixed number of stops/periods (`.`).

The set of valid characters is usually limited to letters, numbers, hyphens (`-`) and underscores (`_`) but this will be specified. Filenames containing forbidden characters or extensions will not be processed.

3.1.4 File processing requirements

FASTQ

- Must be gzipped.
- Must adhere to the FASTQ format.

CSV

- Must be a plain text file with comma-delimited data.
- Must contain two rows: the first will contain the column names and the second will contain the data.
- Must have column names that match the specification exactly.
- Must not have missing data for required fields.
- Must not have invalid data (e.g. `"N/A"`) to circumvent missing data checks.
- Must not contain metadata that contradicts the file name.
- Must use the latest version of the metadata specification.

3.1.5 Metadata specification

The metadata for each project is specified in tables detailing required fields (which must not be empty) and optional fields (which can be left empty).

3.1.6 Project upload buckets

Files should be uploaded to S3 buckets hosted at the `s3.climb.ac.uk` endpoint.

The bucket names are a combination of:

- Project (e.g. `mscape`).
- Site code (e.g. `bham`).
- Platform (e.g. `illumina`).
- A flag that indicates a test (`test`) or production (`prod`) submission.

All files must be placed in the root directory of the submission buckets. Any S3 URI containing a directory will be ignored. *Last modified 2024-03-25 15:29:15+00:00 (ac20a21)*

3.2 PATH-SAFE

3.2.1 PATH-SAFE Uploader Specification

Files to be provided

- A FASTQ 1 file containing the forward sequencing reads.
- A FASTQ 2 file containing the reverse sequencing reads.
- A CSV file containing the metadata associated with sequencing the sample.

File naming convention

The base filenames should be of the form

```
pathsafe.[run_index].[run_id].[extension]
```

where:

- `[run_index]` is an identifier that is unique within a sequencing run, e.g. a sequencing barcode identifier, or a 96-well plate co-ordinate.
- `[run_id]` is the name of the sequencing run as given by the supplier's sequencing instrument (not an internal identifier assigned by the supplier).
- `[extension]` is the file extension indicating the file type.

File name extensions

The extensions (`[extension]`) should be:

- `1.fastq.gz` for the forward FASTQ file.
- `2.fastq.gz` for the reverse FASTQ file.
- `csv` for the CSV metadata file.

Valid characters

The `[run_index]`, `[run_id]` and `[extension]` must contain only:

- Letters (`A-Z`, `a-z`).
- Numbers (`0-9`).
- Hyphens (`-`).
- Underscores (`_`).

Metadata specification

REQUIRED FIELDS

Field	Data type	Description	Restrictions
`biosample_id`	`text`	The sequencing providers identifier for a sample.	• Max length: `50`
`run_index`	`text`	The sequencing provider's identifier for the position of a sample on a run.	• Max length: `50`
`run_id`	`text`	The unique identifier assigned to the run by the sequencing instrument.	• Max length: `100`
`year`	`integer`	Year of sample collected if available or year of sample receipt otherwise.	
`data_steward`	`choice`	Laboratory, organisation or agency that hold the data for the sample.	• Choices: `APHA`, `FSA`, `FSS`, `OTHER`, `PHS`
`source_type`	`choice`	Source of the sample.	• Choices: `animal`, `animal_associated_e`
`country`	`choice`	The country that the sample was collected in, using ISO-3166-1 alpha-2 codes (https://en.wikipedia.org/wiki/List_of_ISO_3166_country_codes), unless within United Kingdom. If so, use ISO-3166-2:GB (https://en.wikipedia.org/wiki/ISO_3166-2:GB).	• Choices: `GB`, `GB-ENG`, `GB-NIR`, `GB-SCT`
`sample_purpose`	`choice`	The purpose of the sample collection.	• Choices: `active_surveillance`, `not_ap` `routine_surveillance`

OPTIONAL FIELDS

Field	Data type	Description	Restrictions
`biosample_source_id`	`text`	Unique identifier for an individual to permit multiple samples from the same individual to be linked.	• Max length: `50`
`sample_accession`	`text`	Sample accession number if sequence is publically available in SRA.	
`enterobase_barcode`	`text`	Sample barcode if sequence is publically available in EnteroBase.	
`collection_date`	`date`	Date of sample collection.	• Input formats: `YYYY-MM` • Output format: `YYYY-MM`
`receipt_date`	`date`	Date of receipt of the sample.	• Input formats: `YYYY-MM` • Output format: `YYYY-MM`
`month`	`integer`	Month of sample collected if available or month of receipt otherwise.	
`sequence_org`	`choice`	Laboratory, organisation or agency the sample has been sequenced by.	• Choices: `APHA`, `FSA`, `FSS`, `PHS`, `SSSCDRL`, `UKHSA`
`sequence_org_other`	`text`	Additional laboratory, organisation or agency the sample has been sequenced by.	• Requires: `sequence_org`
`data_steward_other`	`text`	Additional laboratory, organisation or agency that hold the data for the sample.	• Required when `data_steward` is: `OTHER`

Field	Data type	Description	Restrictions
`county`	`choice`	County that the sample was collected in, using the second level subdivision codes of ISO-3166-2:GB (https://www.iso.org/obp/ui/#iso:code:3166:GB).	• Requires: `country` • Choices: `GB-ABC`, `GB-ABD`, `GB-ABE`, `GB-AGB`, `GB-AGY`, `GB-AND`, `GB-ANN`, `GB-ANS`, `GB-BAS`, `GB-BBD`, `GB-BCP`, `GB-BDF`, `GB-BDG`, `GB-BEN`, `GB-BEX`, `GB-BFS`, `GB-BGE`, `GB-BGW`, `GB-BIR`, `GB-BKM`, ...
`sample_purpose_other`	`text`	Additional purpose of the sample collection.	• Required when `sample_purpose` is: `other`
`sequencing_kit`	`text`	The sequencing kit used.	
`library_kit`	`text`	The library kit used to prep the sample.	
`is_multiplexed`	`bool`	Whether the sample was multiplexed.	

Last modified 2024-03-25 15:29:15+00:00 (ac20a21)

3.2.2 PATH-SAFE Analysis Specification

ANALYSIS FIELDS

Field	Data type	Description	Restrictions
`climb_id`	`text`	Unique identifier for a project record in Onyx.	
`published_date`	`date`	The date the project record was published in Onyx.	• Output format: `iso-8601`
`site`	`choice`	Laboratory, organisation or agency the sample has been submitted by.	• Choices: `APHA` , `FSA` , `FSS` , `PHS` , `SSSCDI`
`biosample_id`	`text`	The sequencing providers identifier for a sample.	
`biosample_source_id`	`text`	Unique identifier for an individual to permit multiple samples from the same individual to be linked.	
`run_id`	`text`	The unique identifier assigned to the run by the sequencing instrument.	
`platform`	`choice`	The platform used to sequence the data.	• Choices: `illumina`
`sample_accession`	`text`	Sample accession number if sequence is publically available in SRA.	
`enterobase_barcode`	`text`	Sample barcode if sequence is publically available in EnteroBase.	
`collection_date`	`date`	Date of sample collection.	• Output format: `YYYY-MM`
`receipt_date`	`date`	Date of receipt of the sample.	• Output format: `YYYY-MM`
`month`	`integer`	Month of sample collected if available or month of receipt otherwise.	
`year`	`integer`	Year of sample collected if available or year of sample receipt otherwise.	
`sequence_org`	`choice`	Laboratory, organisation or agency the sample has been sequenced by.	• Choices: `APHA` , `FSA` , `FSS` , `PHS` , `SSSCDI`
`sequence_org_other`	`text`	Additional laboratory, organisation or agency the sample has been sequenced by.	
`data_steward`	`choice`	Laboratory, organisation or agency that hold the data for the sample.	• Choices: `APHA` , `FSA` , `FSS` , `OTHER` , `PHS`
`data_steward_other`	`text`	Additional laboratory, organisation or agency that hold the data for the sample.	
`source_type`	`choice`	Source of the sample.	• Choices: `animal` , `animal_associated_e`
`country`	`choice`		• Choices: `GB` , `GB-ENG` , `GB-NIR` , `GB-SCT`

Field	Data type	Description	Restrictions
		The country that the sample was collected in, using ISO-3166-1 alpha-2 codes (https://en.wikipedia.org/wiki/List_of_ISO_3166_country_codes), unless within United Kingdom. If so, use ISO-3166-2:GB (https://en.wikipedia.org/wiki/ISO_3166-2:GB).	
`county`	`choice`	County that the sample was collected in, using the second level subdivision codes of ISO-3166-2:GB (https://www.iso.org/obp/ui/#iso:code:3166:GB).	• Choices: `GB-ABC` , `GB-ABD` , `GB-ABE` , `GB-`
`sample_purpose`	`choice`	The purpose of the sample collection.	• Choices: `active_surveillance` , `not_ap` `routine_surveillance`
`sample_purpose_other`	`text`	Additional purpose of the sample collection.	
`sequencing_kit`	`text`	The sequencing kit used.	
`library_kit`	`text`	The library kit used to prep the sample.	
`is_multiplexed`	`bool`	Whether the sample was multiplexed.	
`type_of_sample`	`choice`	Type of sample used to produce the sequence.	• Choices: `genomic`
`assembly`	`text`		
`pathogenwatch_uuid`	`text`		

Last modified 2024-03-25 15:29:15+00:00 (ac20a21)

3.3 mSCAPE

3.3.1 mSCAPE Uploader Specification

Files to be provided

For *paired-end Illumina* sequencing data, suppliers must provide:

- A FASTQ 1 file containing the forward sequencing reads.
- A FASTQ 2 file containing the reverse sequencing reads.
- A CSV file containing the metadata associated with sequencing the sample.

For *ONT* sequencing data, suppliers must provide:

- A FASTQ file containing the sequencing reads.
- A CSV file containing the metadata associated with sequencing the sample.

File naming convention

The base filenames should be of the form

```
mscape.[run_index].[run_id].[extension]
```

where:

- `[run_index]` is an identifier that is unique within a sequencing run, e.g. a sequencing barcode identifier, or a 96-well plate co-ordinate.
- `[run_id]` is the name of the sequencing run as given by the supplier's sequencing instrument (not an internal identifier assigned by the supplier).
- `[extension]` is the file extension indicating the file type.

File name extensions

For *paired-end Illumina* sequencing data, the extensions (`[extension]`) should be:

- `1.fastq.gz` for the forward FASTQ file.
- `2.fastq.gz` for the reverse FASTQ file.
- `csv` for the CSV metadata file.

For *ONT* sequencing data, the extensions (`[extension]`) should be:

- `fastq.gz` for the forward FASTQ file.
- `csv` for the CSV metadata file.

Valid characters

The `[run_index]`, `[run_id]` and `[extension]` must contain only:

- Letters (`A-Z`, `a-z`).
- Numbers (`0-9`).
- Hyphens (`-`).
- Underscores (`_`).

Buckets

Bucket names follow the general convention:

```
mscape-[sequencing_org]-[platform]-[test_flag]
```

Metadata specification

REQUIRED FIELDS

Field	Data type	Description	Restrictions
`biosample_id`	`text`	The sequencing provider's identifier for a sample.	• Max length: `50`
`run_index`	`text`	The sequencing provider's identifier for the position of a sample on a run.	• Max length: `50`
`run_id`	`text`	Unique identifier assigned to the run by the sequencing instrument.	• Max length: `100`
`input_type`	`choice`	The type of input sequenced.	• Choices: `community_standard`, `negative_control`, `positive`
`sample_source`	`choice`	The source from which the sample was collected.	• Choices: `blood`, `environment`, `faecal`, `lower_respiratory` `upper_respiratory`, `urine`
`sample_type`	`choice`	The type of sampling method used.	• Choices: `aspirate`, `bal`, `biopsy`, `other`, `sputum`, `swab`
`spike_in`	`choice`	The type of spike-in used in the run.	• Choices: `none`

At least one of the following fields are required:

Field	Data type	Description	Restrictions
`collection_date`	`date`	The date the sample was collected.	• Input formats: `YYYY-MM`, `YYYY-MM-DD` • Output format: `YYYY-MM-DD`
`received_date`	`date`	The date the sample was received by the sequencing centre (if collection_date unavailable).	• Input formats: `YYYY-MM`, `YYYY-MM-DD` • Output format: `YYYY-MM-DD`

OPTIONAL FIELDS

Field	Data type	Description	Restrictions
`biosample_source_id`	`text`	Unique identifier for an individual to permit multiple samples from the same individual to be linked.	• Max length: `50`
`specimen_type_details`	`choice`	Named control or standard for specimens.	• Required when `input_type` is: `specim` • Choices: `respiratory_infection`
`control_type_details`	`choice`	Named control or standard for positive and negative controls.	• Required when `input_type` is: `positi` • Required when `input_type` is: `negati` • Choices: `water_extraction_control`
`is_approximate_date`	`bool`	The date is approximate e.g. the sample is from a public repository and it is unclear whether the date corresponds to collection or publishing.	• Default: `False`
`batch_id`	`text`	Used to identify samples prepared in the same laboratory batch (e.g. extraction, library and/or sequencing).	• Max length: `100`
`study_id`	`choice`	Used to identify study or if NHS residual sample.	
`study_centre_id`	`text`	Used to identify sequencing centre.	• Max length: `100`
`sequence_purpose`	`choice`	Used to differentiate between clinical or research studies.	• Choices: `clinical`, `research`
`governance_status`	`choice`	Did the patient consent to their sample being used for research purposes or not.	• Default: `no_consent_for_research` • Choices: `consented_for_research`, `no_` `open`
`iso_country`	`choice`	Country that the sample was collected in, using ISO-3166-1 alpha-2 codes (https://en.wikipedia.org/wiki/List_of_ISO_3166_country_codes), unless within United Kingdom. If so, use ISO-3166-2:GB (https://en.wikipedia.org/wiki/ISO_3166-2:GB).	• Choices: `GB`, `GB-ENG`, `GB-NIR`, `GB-SCT`
`iso_region`	`choice`	Region that the sample was collected in, using the second level subdivision codes of ISO-3166-2:GB (https://www.iso.org/obp/ui/#iso:code:3166:GB).	• Requires: `iso_country` • Choices: `GB-ABC`, `GB-ABD`, `GB-ABE`, `GB-` `GB-ANN`, `GB-ANS`, `GB-BAS`, `GB-BBD`, `GB-BC` `BEN`, `GB-BEX`, `GB-BFS`, `GB-BGE`, `GB-BGW`,
`extraction_enrichment_protocol`	`text`	Details of nucleic acid extraction and optional enrichment steps.	
`library_protocol`	`text`	Details of sequencing library construction.	
`sequencing_protocol`	`text`	Details of sequencing.	
`bioinformatics_protocol`	`text`	Detail of initial bioinformatics protocol, for example versions of	

Field	Data type	Description	Restrictions
		basecalling software and models used, any read quality filtering/ trimming employed.	
`dehumanisation_protocol`	`text`	Details of bioinformatics method used for human read removal.	
`is_public_dataset`	`bool`	The sample is from a public dataset. Please only set this after it has been made public.	• Default: `False`
`public_database_name`	`choice`	The public repository where the data is.	• Choices: `ENA` , `SRA`
`public_database_accession`	`text`	The accession for the data in the public database.	

Last modified 2024-03-25 15:29:15+00:00 (ac20a21)

3.3.2 mSCAPE Analysis Specification

ANALYSIS FIELDS

Field	Data type	Description	Restrictions
`climb_id`	`text`	Unique identifier for a project record in Onyx.	
`published_date`	`date`	The date the project record was published in Onyx.	• Output format: `iso-8601`
`site`	`choice`	The site or sequencing centre providing the data.	• Choices: `bham` , `gstt` , `public` , `ukhsa`
`biosample_id`	`text`	The sequencing provider's identifier for a sample.	
`biosample_source_id`	`text`	Unique identifier for an individual to permit multiple samples from the same individual to be linked.	
`run_id`	`text`	Unique identifier assigned to the run by the sequencing instrument.	
`platform`	`choice`	The platform used to sequence the data.	• Choices: `illumina` , `ont`
`input_type`	`choice`	The type of input sequenced.	• Choices: `community_standard` , `negat`
`specimen_type_details`	`choice`	Named control or standard for specimens.	• Choices: `respiratory_infection`
`control_type_details`	`choice`	Named control or standard for positive and negative controls.	• Choices: `water_extraction_control`
`sample_source`	`choice`	The source from which the sample was collected.	• Choices: `blood` , `environment` , `faeca` `upper_respiratory` , `urine`
`sample_type`	`choice`	The type of sampling method used.	• Choices: `aspirate` , `bal` , `biopsy` , `ot`
`spike_in`	`choice`	The type of spike-in used in the run.	• Choices: `none`
`collection_date`	`date`	The date the sample was collected.	• Output format: `YYYY-MM-DD`
`received_date`	`date`	The date the sample was received by the sequencing centre (if collection_date unavailable).	• Output format: `YYYY-MM-DD`
`is_approximate_date`	`bool`	The date is approximate e.g. the sample is from a public repository and it is unclear whether the date corresponds to collection or publishing.	
`batch_id`	`text`	Used to identify samples prepared in the same laboratory batch (e.g. extraction, library and/or sequencing).	
`study_id`	`choice`	Used to identify study or if NHS residual sample.	
`study_centre_id`	`text`	Used to identify sequencing centre.	

Field	Data type	Description	Restrictions
`sequence_purpose`	`choice`	Used to differentiate between clinical or research studies.	• Choices: `clinical` , `research`
`governance_status`	`choice`	Did the patient consent to their sample being used for research purposes or not.	• Choices: `consented_for_research` , `n`
`iso_country`	`choice`	Country that the sample was collected in, using ISO-3166-1 alpha-2 codes (https://en.wikipedia.org/wiki/List_of_ISO_3166_country_codes), unless within United Kingdom. If so, use ISO-3166-2:GB (https://en.wikipedia.org/wiki/ISO_3166-2:GB).	• Choices: `GB` , `GB-ENG` , `GB-NIR` , `GB-SC`
`iso_region`	`choice`	Region that the sample was collected in, using the second level subdivision codes of ISO-3166-2:GB (https://www.iso.org/obp/ui/#iso:code:3166:GB).	• Choices: `GB-ABC` , `GB-ABD` , `GB-ABE` , `G` `BEN` , `GB-BEX` , `GB-BFS` , `GB-BGE` , `GB-BGW`
`extraction_enrichment_protocol`	`text`	Details of nucleic acid extraction and optional enrichment steps.	
`library_protocol`	`text`	Details of sequencing library construction.	
`sequencing_protocol`	`text`	Details of sequencing.	
`bioinformatics_protocol`	`text`	Detail of initial bioinformatics protocol, for example versions of basecalling software and models used, any read quality filtering/trimming employed.	
`dehumanisation_protocol`	`text`	Details of bioinformatics method used for human read removal.	
`is_public_dataset`	`bool`	The sample is from a public dataset. Please only set this after it has been made public.	
`public_database_name`	`choice`	The public repository where the data is.	• Choices: `ENA` , `SRA`
`public_database_accession`	`text`	The accession for the data in the public database.	
`ingest_report`	`text`	HTML report summarising the read profile and taxa identified.	
`taxon_reports`	`text`	Folder of all classification output files.	
`human_filtered_reads_1`	`text`	Compressed FASTQ of input reads that have been filtered for human reads.	
`human_filtered_reads_2`	`text`		

Field	Data type	Description	Restrictions
		Compressed FASTQ of input reads that have been filtered for human reads.	
`unclassified_reads_1`	`text`	Compressed FASTQ of input reads which could not be classified.	
`unclassified_reads_2`	`text`	Compressed FASTQ of input reads which could not be classified.	
`viral_reads_1`	`text`	Compressed FASTQ of input reads which were classified as viral.	
`viral_reads_2`	`text`	Compressed FASTQ of input reads which were classified as viral.	
`viral_and_unclassified_reads_1`	`text`	Compressed FASTQ of input reads which were classified as viral or were unclassified.	
`viral_and_unclassified_reads_2`	`text`	Compressed FASTQ of input reads which were classified as viral or were unclassified.	
`classifier`	`choice`	The classifier used.	• Choices: `Kraken2`
`classifier_version`	`text`	Version of the classifier used.	
`classifier_db`	`choice`	Database used for read classification.	• Choices: `PlusPF`
`classifier_db_date`	`date`	Date classifier database was produced.	• Output format: `YYYY-MM-DD`
`ncbi_taxonomy_date`	`date`	Date that the NCBI taxonomy dump was produced.	• Output format: `YYYY-MM-DD`
`scylla_version`	`text`	Version of the scylla pipeline used.	
`taxa_files`	`relation`	Table of all species level taxa extracted.	
`taxa_files.taxon_id`	`integer`	The NCBI taxonomy id associated with the taxa.	
`taxa_files.human_readable`	`text`	A human readable name for the taxa.	
`taxa_files.n_reads`	`integer`	The number of reads extracted for the taxa.	
`taxa_files.avg_quality`	`decimal`	The mean quality of reads extracted for the taxa.	
`taxa_files.mean_len`	`decimal`	The mean length of reads extracted for the taxa.	
`taxa_files.rank`	`choice`	The rank of the taxa.	• Choices: `C`, `D`, `F`, `G`, `K`, `O`, `P`, `R`,

Field	Data type	Description	Restrictions
`taxa_files.fastq_1`	`text`	Compressed FASTQ of extracted reads for the taxa.	
`taxa_files.fastq_2`	`text`	Compressed FASTQ of extracted reads for the taxa.	
`classifier_calls`	`relation`	Table summarising the NCBI taxonomy ids, counts and ranks of all taxa found by the classifier.	
`classifier_calls.taxon_id`	`integer`	The NCBI taxonomy id associated with the taxa.	
`classifier_calls.human_readable`	`text`	A human readable name for the taxa.	
`classifier_calls.percentage`	`decimal`	The percentage of the (dehumanised) sample that the taxa represents.	
`classifier_calls.count_descendants`	`integer`	The number of reads mapping to this taxa and all descendant taxa.	
`classifier_calls.count_direct`	`integer`	The number of reads mapping directly to the taxa.	
`classifier_calls.rank`	`choice`	The rank of the taxa.	• Choices: `C`, `D`, `F`, `G`, `K`, `O`, `P`, `R`,
`classifier_calls.raw_rank`	`text`	The rank of the taxa including an intermediate grading.	

Last modified 2024-03-25 15:29:15+00:00 (ac20a21)