Labelbox Python API reference

Release 2.4

unknown

Jan 25, 2021

CONTENTS:

1 Client 1

2 AssetMetadata 5

3 Benchmark 7

4 BulkImportRequest 9

5 DataRow 11

6 Dataset 13

7 Label 15

8 LabelingFrontend 17

9 LabelingFrontendOptions 19

10 LabelingParameterOverride 21

11 Ontology 23

12 Organization 25

13 Prediction 27

14 PredictionModel 29

15 Project 31

16 Review 35

17 Task 37

18 User 39

19 Webhook 41

20 Exceptions 43

21 Pagination 45

22 Enums 47

Python Module Index 49

Index 51

CHAPTER

ONE

CLIENT

class labelbox.client.**Client**(*api_key=None*, *endpoint='https://api.labelbox.com/graphql'*)

Bases: object

A Labelbox client.

Contains info necessary for connecting to a Labelbox server (URL, authentication key). Provides functions for querying and creating top-level data objects (Projects, Datasets).

__init__(*api_key=None*, *endpoint='https://api.labelbox.com/graphql'*)

Creates and initializes a Labelbox Client.

Logging is defaulted to level WARNING. To receive more verbose output to console, update *logging.level* to the appropriate level.

```
>>> import logger
>>> logging.basicConfig(level = logging.INFO)
>>> client = Client("<APIKEY>")
```

Parameters

- **api_key** (*str*) – API key. If None, the key is obtained from the "LABELBOX_API_KEY" environment variable.
- **endpoint** (*str*) – URL of the Labelbox server to connect to.

Raises ***labelbox.exceptions.AuthenticationError*** – If no *api_key* is provided as an argument or via the environment variable.

create_dataset(***kwargs*)

Creates a Dataset object on the server.

Attribute values are passed as keyword arguments.

```
>>> project = client.get_project("<project_uid>")
>>> dataset = client.create_dataset(name="<dataset_name>", projects=project)
```

Parameters ****kwargs** – Keyword arguments with Dataset attribute values.

Returns A new Dataset object.

Raises ***InvalidAttributeError*** – If the Dataset type does not contain any of the attribute names given in kwargs.

create_project(***kwargs*)

Creates a Project object on the server.

Attribute values are passed as keyword arguments.

```
>>> project = client.create_project(name="<project_name>", description="
↪<project_description>")
```

Parameters ****kwargs** – Keyword arguments with Project attribute values.

Returns A new Project object.

Raises ***InvalidAttributeError*** – If the Project type does not contain any of the attribute names given in kwargs.

execute(*query*, *params=None*, *timeout=10.0*)

Sends a request to the server for the execution of the given query.

Checks the response for errors and wraps errors in appropriate *labelbox.exceptions.LabelboxError* subtypes.

Parameters

- **query** (*str*) – The query to execute.
- **params** (*dict*) – Query parameters referenced within the query.
- **timeout** (*float*) – Max allowed time for query execution, in seconds.

Returns dict, parsed JSON response.

Raises

- ***labelbox.exceptions.AuthenticationError*** – If authentication failed.
- ***labelbox.exceptions.InvalidQueryError*** – If *query* is not syntactically or semantically valid (checked server-side).
- ***labelbox.exceptions.ApiLimitError*** – If the server API limit was exceeded. See "How to import data" in the online documentation to see API limits.
- ***labelbox.exceptions.TimeoutError*** – If response was not received in *timeout* seconds.
- ***labelbox.exceptions.NetworkError*** – If an unknown error occurred most likely due to connection issues.
- ***labelbox.exceptions.LabelboxError*** – If an unknown error of any kind occurred.

get_dataset(*dataset_id*)

Gets a single Dataset with the given ID.

```
>>> dataset = client.get_dataset("<dataset_id>")
```

Parameters **dataset_id** (*str*) – Unique ID of the Dataset.

Returns The sought Dataset.

Raises ***labelbox.exceptions.ResourceNotFoundError*** – If there is no Dataset with the given ID.

get_datasets(*where=None*)

Fetches one or more datasets.

```
>>> datasets = client.get_datasets(where=(Dataset.name == "<dataset_name>") &␣
↪(Dataset.description == "<dataset_description>"))
```

Parameters **where** (*Comparison, LogicalOperation or None*) – The *where* clause for filtering.

Returns An iterable of Datasets (typically a PaginatedCollection).

get_labeling_frontends(*where=None*)

Fetches all the labeling frontends.

```
>>> frontend = client.get_labeling_frontends(where=LabelingFrontend.name ==
↪"Editor")
```

Parameters **where** (*Comparison, LogicalOperation or None*) – The *where* clause for filtering.

Returns An iterable of LabelingFrontends (typically a PaginatedCollection).

get_organization()

Gets the Organization DB object of the current user.

```
>>> organization = client.get_organization()
```

get_project(*project_id*)

Gets a single Project with the given ID.

```
>>> project = client.get_project("<project_id>")
```

Parameters **project_id** (*str*) – Unique ID of the Project.

Returns The sought Project.

Raises ***labelbox.exceptions.ResourceNotFoundError*** – If there is no Project with the given ID.

get_projects(*where=None*)

Fetches all the projects the user has access to.

```
>>> projects = client.get_projects(where=(Project.name == "<project_name>") &␣
↪(Project.description == "<project_description>"))
```

Parameters **where** (*Comparison, LogicalOperation or None*) – The *where* clause for filtering.

Returns An iterable of Projects (typically a PaginatedCollection).

get_user()

Gets the current User database object.

```
>>> user = client.get_user()
```

CHAPTER

TWO

ASSETMETADATA

class labelbox.schema.asset_metadata.**AssetMetadata**(*client*, *field_values*)

Bases: labelbox.orm.db_object.DbObject

Asset metadata (AKA Attachments) provides extra context about an asset while labeling.

meta_type

IMAGE, VIDEO, TEXT, or IMAGE_OVERLAY

Type str

meta_value

URL to an external file or a string of text

Type str

CHAPTER

THREE

BENCHMARK

class labelbox.schema.benchmark.**Benchmark**(*client*, *field_values*)

Bases: labelbox.orm.db_object.DbObject

Represents a benchmark label.

The Benchmarks tool works by interspersing data to be labeled, for which there is a benchmark label, to each person labeling. These labeled data are compared against their respective benchmark and an accuracy score between 0 and 100 percent is calculated.

created_at

Type datetime

last_activity

Type datetime

average_agreement

Type float

completed_count

Type int

created_by

ToOne relationship to User

Type Relationship

reference_label

ToOne relationship to Label

Type Relationship

CHAPTER

FOUR

BULKIMPORTREQUEST

class labelbox.schema.bulk_import_request.**BulkImportRequest**(*client*, *field_values*)

Bases: labelbox.orm.db_object.DbObject

Represents the import job when importing annotations.

name

Type str

state

FAILED, RUNNING, or FINISHED (Refers to the whole import job)

Type Enum

input_file_url

URL to your web-hosted NDJSON file

Type str

error_file_url

NDJSON that contains error messages for failed annotations

Type str

status_file_url

NDJSON that contains status for each annotation

Type str

created_at

UTC timestamp for date BulkImportRequest was created

Type datetime

project

ToOne relationship to Project

Type Relationship

created_by

ToOne relationship to User

Type Relationship

refresh() → None

Synchronizes values of all fields with the database.

wait_until_done(*sleep_time_seconds: int = 30*) → None

Blocks import job until certain conditions are met.

Blocks until the BulkImportRequest.state changes either to *BulkImportRequestState.FINISHED* or *BulkImportRequestState.FAILED*, periodically refreshing object's state.

Parameters **sleep_time_seconds** (*str*) – a time to block between subsequent API calls

CHAPTER

FIVE

DATAROW

class labelbox.schema.data_row.**DataRow**(**args*, ***kwargs*)

Bases: labelbox.orm.db_object.DbObject, labelbox.orm.db_object.Updateable, labelbox.orm.db_object.BulkDeletable

Internal Labelbox representation of a single piece of data (e.g. image, video, text).

external_id
User-generated file name or identifier

Type str

row_data
Paths to local files are uploaded to Labelbox's server. Otherwise, it's treated as an external URL.

Type str

updated_at

Type datetime

created_at

Type datetime

dataset
ToOne relationship to Dataset

Type Relationship

created_by
ToOne relationship to User

Type Relationship

organization
ToOne relationship to Organization

Type Relationship

labels
ToMany relationship to Label

Type Relationship

metadata
ToMany relationship to AssetMetadata

Type Relationship

predictions
ToMany relationship to Prediction

Type Relationship

static bulk_delete(*data_rows*)
Deletes all the given DataRows.

Parameters **data_rows** (*list of DataRow*) – The DataRows to delete.

create_metadata(*meta_type*, *meta_value*)
Attaches asset metadata to a DataRow.

```
>>> datarow.create_metadata("TEXT", "This is a text message")
```

Parameters

- **meta_type** (*str*) – Asset metadata type, must be one of: VIDEO, IMAGE, TEXT.
- **meta_value** (*str*) – Asset metadata value.

Returns *AssetMetadata* DB object.

CHAPTER

SIX

DATASET

class labelbox.schema.dataset.**Dataset**(*client*, *field_values*)

Bases: labelbox.orm.db_object.DbObject, labelbox.orm.db_object.Updateable, labelbox.orm.db_object.Deletable

A Dataset is a collection of DataRows.

name

Type str

description

Type str

updated_at

Type datetime

created_at

Type datetime

projects

ToMany relationship to Project

Type Relationship

data_rows

ToMany relationship to DataRow

Type Relationship

created_by

ToOne relationship to User

Type Relationship

organization

ToOne relationship to Organization

Type Relationship

create_data_row(***kwargs*)

Creates a single DataRow belonging to this dataset.

```
>>> dataset.create_data_row(row_data="http://my_site.com/photos/img_01.jpg")
```

Parameters ****kwargs** – Key-value arguments containing new *DataRow* data. At a minimum, must contain *row_data*.

Raises

- ***InvalidQueryError*** – If *DataRow.row_data* field value is not provided in *kwargs*.
- ***InvalidAttributeError*** – in case the DB object type does not contain any of the field names given in *kwargs*.

create_data_rows(*items*)

Creates multiple DataRow objects based on the given *items*.

Each element in *items* can be either a *str* or a *dict*. If it is a *str*, then it is interpreted as a local file path. The file is uploaded to Labelbox and a DataRow referencing it is created. If an item is a *dict*, then it should map *DataRow* fields (or their names) to values. At the minimum an *item* passed as a *dict* must contain a *DataRow.row_data* key and value.

```
>>> dataset.create_data_rows([
>>>     {DataRow.row_data:"http://my_site.com/photos/img_01.jpg"},
>>>     "path/to/file2.jpg"
>>>     ])
```

Parameters **items** (*iterable of (dict or str)*) – See above for details.

Returns Task representing the data import on the server side. The Task can be used for inspecting task progress and waiting until it's done.

Raises

- ***InvalidQueryError*** – If the *items* parameter does not conform to the specification above or if the server did not accept the DataRow creation request (unknown reason).
- ***ResourceNotFoundError*** – If unable to retrieve the Task for the import process. This could imply that the import failed.
- ***InvalidAttributeError*** – If there are fields in *items* not valid for a DataRow.

data_row_for_external_id(*external_id*)

Convenience method for getting a single *DataRow* belonging to this *Dataset* that has the given *external_id*.

Parameters **external_id** (*str*) – External ID of the sought *DataRow*.

Returns A single *DataRow* with the given ID.

Raises ***labelbox.exceptions.ResourceNotFoundError*** – If there is no *DataRow* in this *DataSet* with the given external ID, or if there are multiple *DataRows* for it.

CHAPTER

SEVEN

LABEL

class labelbox.schema.label.**Label**(**args*, ***kwargs*)

Bases: labelbox.orm.db_object.DbObject, labelbox.orm.db_object.Updateable, labelbox.orm.db_object.BulkDeletable

Label represents an assessment on a DataRow. For example one label could contain 100 bounding boxes (annotations).

label

Type str

seconds_to_label

Type float

agreement

Type float

benchmark_agreement

Type float

is_benchmark_reference

Type bool

project

ToOne relationship to Project

Type Relationship

data_row

ToOne relationship to DataRow

Type Relationship

reviews

ToMany relationship to Review

Type Relationship

created_by

ToOne relationship to User

Type Relationship

static bulk_delete(*labels*)

Deletes all the given Labels.

Parameters **labels** (*list of Label*) – The Labels to delete.

create_benchmark()
Creates a Benchmark for this Label.

Returns The newly created Benchmark.

create_review(**kwargs)
Creates a Review for this label.

Parameters ****kwargs** – Review attributes. At a minimum, a *Review.score* field value must be provided.

CHAPTER

EIGHT

LABELINGFRONTEND

class labelbox.schema.labeling_frontend.**LabelingFrontend**(*client*, *field_values*)

Bases: labelbox.orm.db_object.DbObject

Label editor.

Represents an HTML / JavaScript UI that is used to generate labels. "Editor" is the default Labeling Frontend that comes in every organization. You can create new labeling frontends for an organization.

name

Type str

description

Type str

iframe_url_path

Type str

projects

ToMany relationship to Project

Type Relationship

CHAPTER

NINE

LABELINGFRONTENDOPTIONS

class labelbox.schema.labeling_frontend.**LabelingFrontendOptions**(*client*, *field_values*)

Bases: labelbox.orm.db_object.DbObject

Label interface options.

customization_options

Type str

project

ToOne relationship to Project

Type Relationship

labeling_frontend

ToOne relationship to LabelingFrontend

Type Relationship

organization

ToOne relationship to Organization

Type Relationship

CHAPTER

TEN

LABELINGPARAMETEROVERRIDE

class labelbox.schema.project.**LabelingParameterOverride**(*client*, *field_values*)

Bases: labelbox.orm.db_object.DbObject

Customizes the order of assets in the label queue.

priority

A prioritization score.

Type int

number_of_labels

Number of times an asset should be labeled.

Type int

CHAPTER

ELEVEN

ONTOLOGY

class labelbox.schema.ontology.**Ontology**(**args*, ***kwargs*)
Bases: labelbox.orm.db_object.DbObject

An ontology specifies which tools and classifications are available to a project. This is read only for now.

name

Type str

description

Type str

updated_at

Type datetime

created_at

Type datetime

normalized

Type json

object_schema_count

Type int

classification_schema_count

Type int

projects
ToMany relationship to Project

Type Relationship

created_by
ToOne relationship to User

Type Relationship

classifications() → List[labelbox.schema.ontology.Classification]
Get list of classifications in an Ontology.

tools() → List[labelbox.schema.ontology.Tool]
Get list of tools (AKA objects) in an Ontology.

CHAPTER

TWELVE

ORGANIZATION

class labelbox.schema.organization.**Organization**(**args*, ***kwargs*)

Bases: labelbox.orm.db_object.DbObject

An Organization is a group of Users.

It is associated with data created by Users within that Organization. Typically all Users within an Organization have access to data created by any User in the same Organization.

updated_at

Type datetime

created_at

Type datetime

name

Type str

users

ToMany relationship to User

Type Relationship

projects

ToMany relationship to Project

Type Relationship

webhooks

ToMany relationship to Webhook

Type Relationship

CHAPTER

THIRTEEN

PREDICTION

class labelbox.schema.prediction.**Prediction**(*client*, *field_values*)

Bases: labelbox.orm.db_object.DbObject

A prediction created by a PredictionModel. Legacy editor only.

Refer to BulkImportRequest if using the new Editor.

updated_at

Type datetime

created_at

Type datetime

label

Type str

agreement

Type float

organization

ToOne relationship to Organization

Type Relationship

prediction_model

ToOne relationship to PredictionModel

Type Relationship

data_row

ToOne relationship to DataRow

Type Relationship

project

ToOne relationship to Project

Type Relationship

CHAPTER

FOURTEEN

PREDICTIONMODEL

class labelbox.schema.prediction.**PredictionModel**(*client*, *field_values*)

Bases: labelbox.orm.db_object.DbObject

A PredictionModel creates a Prediction. Legacy editor only.

Refer to BulkImportRequest if using the new Editor.

updated_at

Type datetime

created_at

Type datetime

name

Type str

slug

Type str

version

Type int

created_by

ToOne relationship to User

Type Relationship

organization

ToOne relationship to Organization

Type Relationship

CHAPTER

FIFTEEN

PROJECT

class labelbox.schema.project.**Project**(*client*, *field_values*)

Bases: labelbox.orm.db_object.DbObject, labelbox.orm.db_object.Updateable, labelbox.orm.db_object.Deletable

A Project is a container that includes a labeling frontend, an ontology, datasets and labels.

name

Type str

description

Type str

updated_at

Type datetime

created_at

Type datetime

setup_complete

Type datetime

last_activity_time

Type datetime

auto_audit_number_of_labels

Type int

auto_audit_percentage

Type float

datasets

ToMany relationship to Dataset

Type Relationship

created_by

ToOne relationship to User

Type Relationship

organization

ToOne relationship to Organization

Type Relationship

reviews
ToMany relationship to Review

Type Relationship

labeling_frontend
ToOne relationship to LabelingFrontend

Type Relationship

labeling_frontend_options
ToMany relationship to LabelingFrontendOptions

Type Relationship

labeling_parameter_overrides
ToMany relationship to LabelingParameterOverride

Type Relationship

webhooks
ToMany relationship to Webhook

Type Relationship

benchmarks
ToMany relationship to Benchmark

Type Relationship

active_prediction_model
ToOne relationship to PredictionModel

Type Relationship

predictions
ToMany relationship to Prediction

Type Relationship

ontology
ToOne relationship to Ontology

Type Relationship

create_label(***kwargs*)
Creates a label on a Legacy Editor project. Not supported in the new Editor.

Parameters ****kwargs** – Label attributes. At minimum, the label *DataRow*.

create_prediction(*label*, *data_row*, *prediction_model=None*)
Creates a Prediction within a Legacy Editor Project. Not supported in the new Editor.

Parameters

- **label** (*str*) – The *label* field of the new Prediction.
- **data_row** (DataRow) – The DataRow for which the Prediction is created.
- **prediction_model** (*PredictionModel or None*) – The PredictionModel within which the new Prediction is created. If None then this Project's active_prediction_model is used.

Returns A newly created Prediction.

Raises **labelbox.excepions.InvalidQueryError** – if given *prediction_model* is None and this Project's active_prediction_model is also None.

create_prediction_model (*name*, *version*)
Creates a PredictionModel connected to a Legacy Editor Project.

Parameters

- **name** (*str*) – The new PredictionModel's name.
- **version** (*int*) – The new PredictionModel's version.

Returns A newly created PredictionModel.

enable_model_assisted_labeling (*toggle: bool = True*) → bool
Turns model assisted labeling either on or off based on input

Parameters **toggle** (*bool*) – True or False boolean

Returns True if toggled on or False if toggled off

export_labels (*timeout_seconds=60*)
Calls the server-side Label exporting that generates a JSON payload, and returns the URL to that payload.

Will only generate a new URL at a max frequency of 30 min.

Parameters **timeout_seconds** (*float*) – Max waiting time, in seconds.

Returns URL of the data file with this Project's labels. If the server didn't generate during the *timeout_seconds* period, None is returned.

extend_reservations (*queue_type*)
Extends all the current reservations for the current user on the given queue type.

Parameters **queue_type** (*str*) – Either "LabelingQueue" or "ReviewQueue"

Returns int, the number of reservations that were extended.

labeler_performance ()
Returns the labeler performances for this Project.

Returns A PaginatedCollection of LabelerPerformance objects.

labels (*datasets=None*, *order_by=None*)
Custom relationship expansion method to support limited filtering.

Parameters

- **datasets** (*iterable of Dataset*) – Optional collection of Datasets whose Labels are sought. If not provided, all Labels in this Project are returned.
- **order_by** (*None or (Field, Field.Order)*) – Ordering clause.

review_metrics (*net_score*)
Returns this Project's review metrics.

Parameters **net_score** (*None or Review.NetScore*) – Indicates desired metric.

Returns int, aggregation count of reviews for given *net_score*.

set_labeling_parameter_overrides (*data*)
Adds labeling parameter overrides to this project.

```
>>> project.set_labeling_parameter_overrides([
>>>     (data_row_1, 2, 3), (data_row_2, 1, 4)])
```

Parameters **data** (*iterable*) – An iterable of tuples. Each tuple must contain (DataRow, priority, numberOfLabels) for the new override.

Returns bool, indicates if the operation was a success.

setup(*labeling_frontend*, *labeling_frontend_options*)
Finalizes the Project setup.

Parameters

- **labeling_frontend** (LabelingFrontend) – Which UI to use to label the data.
- **labeling_frontend_options** (*dict or str*) – Labeling frontend options, a.k.a. project ontology. If given a *dict* it will be converted to *str* using *json.dumps*.

unset_labeling_parameter_overrides(*data_rows*)
Removes labeling parameter overrides to this project.

Parameters **data_rows** (*iterable*) – An iterable of DataRows.

Returns bool, indicates if the operation was a success.

upload_annotations(*name: str*, *annotations: Union[str, pathlib.Path, Iterable[dict]]*) → *labelbox.schema.bulk_import_request.BulkImportRequest*
Uploads annotations to a new Editor project.

Parameters

- **name** (*str*) – name of the BulkImportRequest job
- **annotations** (*str or Path or Iterable*) – url that is publicly accessible by Labelbox containing an ndjson file OR local path to an ndjson file OR iterable of annotation rows

Returns BulkImportRequest

upsert_review_queue(*quota_factor*)
Reinitiates the review queue for this project.

Parameters **quota_factor** (*float*) – Which part (percentage) of the queue to reinitiate. Between 0 and 1.

CHAPTER

SIXTEEN

REVIEW

class labelbox.schema.review.**Review**(*client*, *field_values*)

Bases: labelbox.orm.db_object.DbObject, labelbox.orm.db_object.Deletable, labelbox.orm.db_object.Updateable

Reviewing labeled data is a collaborative quality assurance technique.

A Review object indicates the quality of the assigned Label. The aggregated review numbers can be obtained on a Project object.

created_at

Type datetime

updated_at

Type datetime

score

Type float

created_by

ToOne relationship to User

Type Relationship

organization

ToOne relationship to Organization

Type Relationship

project

ToOne relationship to Project

Type Relationship

label

ToOne relationship to Label

Type Relationship

class NetScore(*value*)

Bases: enum.Enum

Negative, Zero, or Positive.

CHAPTER

SEVENTEEN

TASK

class labelbox.schema.task.**Task**(*client*, *field_values*)
Bases: labelbox.orm.db_object.DbObject

Represents a server-side process that might take a longer time to process. Allows the Task state to be updated and checked on the client side.

updated_at

Type datetime

created_at

Type datetime

name

Type str

status

Type str

completion_percentage

Type float

created_by
ToOne relationship to User

Type Relationship

organization
ToOne relationship to Organization

Type Relationship

refresh()
Refreshes Task data from the server.

wait_till_done(*timeout_seconds=60*)
Waits until the task is completed. Periodically queries the server to update the task attributes.

Parameters **timeout_seconds** (*float*) – Maximum time this method can block, in seconds. Defaults to one minute.

CHAPTER

EIGHTEEN

USER

class labelbox.schema.user.**User**(*client*, *field_values*)
Bases: labelbox.orm.db_object.DbObject

A User is a registered Labelbox user (for example you) associated with data they create or import and an Organization they belong to.

updated_at

Type datetime

created_at

Type datetime

email

Type str

name

Type str

nickname

Type str

intercom_hash

Type str

picture

Type str

is_viewer

Type bool

is_external_viewer

Type bool

organization
ToOne relationship to Organization

Type Relationship

created_tasks
ToMany relationship to Task

Type Relationship

projects
ToMany relationship to Project

Type Relationship

CHAPTER

NINETEEN

WEBHOOK

class labelbox.schema.webhook.**Webhook**(*client*, *field_values*)
Bases: labelbox.orm.db_object.DbObject, labelbox.orm.db_object.Updateable

Represents a server-side rule for sending notifications to a web-server whenever one of several predefined actions happens within a context of a Project or an Organization.

updated_at

Type datetime

created_at

Type datetime

url

Type str

topics
LABEL_CREATED, LABEL_UPDATED, LABEL_DELETED

Type str

status
ACTIVE, INACTIVE, REVOKED

Type str

static create(*client*, *topics*, *url*, *secret*, *project*)
Creates a Webhook.

Parameters

- **client** (*Client*) – The Labelbox client used to connect to the server.
- **topics** (*list of str*) – A list of topics this Webhook should get notifications for.
- **url** (*str*) – The URL to which notifications should be sent by the Labelbox server.
- **secret** (*str*) – A secret key used for signing notifications.
- **project** (*Project or None*) – The project for which notifications should be sent. If None notifications are sent for all events in your organization.

Returns A newly created Webhook.

update(*topics=None*, *url=None*, *status=None*)
Updates this Webhook.

Parameters

- **topics** (*list of str*) – The new topics value, optional.

- **url** (*str*) – The new URL value, optional.
- **status** (*str*) – The new status value, optional.

CHAPTER

TWENTY

EXCEPTIONS

exception labelbox.exceptions.**ApiLimitError**(*message*, *cause=None*)
Bases: *labelbox.exceptions.LabelboxError*

Raised when the user performs too many requests in a short period of time.

exception labelbox.exceptions.**AuthenticationError**(*message*, *cause=None*)
Bases: *labelbox.exceptions.LabelboxError*

Raised when an API key fails authentication.

exception labelbox.exceptions.**AuthorizationError**(*message*, *cause=None*)
Bases: *labelbox.exceptions.LabelboxError*

Raised when a user is unauthorized to perform the given request.

exception labelbox.exceptions.**InternalServerError**(*message*, *cause=None*)
Bases: *labelbox.exceptions.LabelboxError*

Nondescript prisma or 502 related errors.

Meant to be retryable.

TODO: these errors need better messages from platform

exception labelbox.exceptions.**InvalidAttributeError**(*db_object_type*, *field*)
Bases: *labelbox.exceptions.LabelboxError*

Raised when a field (name or Field instance) is not valid or found for a specific DB object type.

exception labelbox.exceptions.**InvalidQueryError**(*message*, *cause=None*)
Bases: *labelbox.exceptions.LabelboxError*

Indicates a malconstructed or unsupported query (either by GraphQL in general or by Labelbox specifically). This can be the result of either client or server side query validation.

exception labelbox.exceptions.**LabelboxError**(*message*, *cause=None*)
Bases: Exception

Base class for exceptions.

exception labelbox.exceptions.**MalformedQueryException**
Bases: Exception

Raised when the user submits a malformed query.

exception labelbox.exceptions.**NetworkError**(*cause*)
Bases: *labelbox.exceptions.LabelboxError*

Raised when an HTTPError occurs.

exception labelbox.exceptions.**ResourceNotFoundError**(*db_object_type*, *params*)
Bases: *labelbox.exceptions.LabelboxError*

Exception raised when a given resource is not found.

exception labelbox.exceptions.**TimeoutError**(*message*, *cause=None*)
Bases: *labelbox.exceptions.LabelboxError*

Raised when a request times-out.

exception labelbox.exceptions.**UuidError**(*message*, *cause=None*)
Bases: *labelbox.exceptions.LabelboxError*

Raised when there are repeat Uuid's in bulk import request.

exception labelbox.exceptions.**ValidationFailedError**(*message*, *cause=None*)
Bases: *labelbox.exceptions.LabelboxError*

Exception raised for when a GraphQL query fails validation (query cost, etc.) E.g. a query that is too expensive, or depth is too deep.

CHAPTER

TWENTYONE

PAGINATION

class labelbox.pagination.**PaginatedCollection**(*client*, *query*, *params*, *dereferencing*, *obj_class*)

Bases: object

An iterable collection of database objects (Projects, Labels, etc...).

Implements automatic (transparent to the user) paginated fetching during iteration. Intended for use by library internals and not by the end user. For a list of attributes see __init__(...) documentation. The params of __init__ map exactly to object attributes.

__init__(*client*, *query*, *params*, *dereferencing*, *obj_class*)

Creates a PaginatedCollection.

Parameters

- **client** (*labelbox.Client*) – the client used for fetching data from DB.
- **query** (*str*) – Base query used for pagination. It must contain two '%d' placeholders, the first for pagination 'skip' clause and the second for the 'first' clause.
- **params** (*dict*) – Query parameters.
- **dereferencing** (*iterable*) – An iterable of str defining the keypath that needs to be dereferenced in the query result in order to reach the paginated objects of interest.
- **obj_class** (*type*) – The class of object to be instantiated with each dict containing db values.

CHAPTER

TWENTYTWO

ENUMS

class labelbox.schema.enums.**BulkImportRequestState**(*value*)

Bases: enum.Enum

State of the import job when importing annotations (RUNNING, FAILED, or FINISHED).

PYTHON MODULE INDEX

l

labelbox.client, 1
labelbox.exceptions, 43
labelbox.pagination, 45
labelbox.schema.asset_metadata, 5
labelbox.schema.benchmark, 7
labelbox.schema.bulk_import_request, 9
labelbox.schema.data_row, 11
labelbox.schema.dataset, 13
labelbox.schema.enums, 47
labelbox.schema.label, 15
labelbox.schema.labeling_frontend, 17
labelbox.schema.ontology, 23
labelbox.schema.organization, 25
labelbox.schema.prediction, 27
labelbox.schema.project, 31
labelbox.schema.review, 35
labelbox.schema.task, 37
labelbox.schema.user, 39
labelbox.schema.webhook, 41

INDEX

Symbols

__init__() (*labelbox.client.Client method*), 1
__init__() (*labelbox.pagination.PaginatedCollection method*), 45

A

active_prediction_model (*labelbox.schema.project.Project attribute*), 32
agreement (*labelbox.schema.label.Label attribute*), 15
agreement (*labelbox.schema.prediction.Prediction attribute*), 27
ApiLimitError, 43
AssetMetadata (*class in labelbox.schema.asset_metadata*), 5
AuthenticationError, 43
AuthorizationError, 43
auto_audit_number_of_labels (*labelbox.schema.project.Project attribute*), 31
auto_audit_percentage (*labelbox.schema.project.Project attribute*), 31
average_agreement (*labelbox.schema.benchmark.Benchmark attribute*), 7

B

Benchmark (*class in labelbox.schema.benchmark*), 7
benchmark_agreement (*labelbox.schema.label.Label attribute*), 15
benchmarks (*labelbox.schema.project.Project attribute*), 32
bulk_delete() (*labelbox.schema.data_row.DataRow static method*), 12
bulk_delete() (*labelbox.schema.label.Label static method*), 15
BulkImportRequest (*class in labelbox.schema.bulk_import_request*), 9
BulkImportRequestState (*class in labelbox.schema.enums*), 47

C

classification_schema_count (*labelbox.schema.ontology.Ontology attribute*), 23
classifications() (*labelbox.schema.ontology.Ontology method*), 23
Client (*class in labelbox.client*), 1
completed_count (*labelbox.schema.benchmark.Benchmark attribute*), 7
completion_percentage (*labelbox.schema.task.Task attribute*), 37
create() (*labelbox.schema.webhook.Webhook static method*), 41
create_benchmark() (*labelbox.schema.label.Label method*), 15
create_data_row() (*labelbox.schema.dataset.Dataset method*), 13
create_data_rows() (*labelbox.schema.dataset.Dataset method*), 14
create_dataset() (*labelbox.client.Client method*), 1
create_label() (*labelbox.schema.project.Project method*), 32
create_metadata() (*labelbox.schema.data_row.DataRow method*), 12
create_prediction() (*labelbox.schema.project.Project method*), 32
create_prediction_model() (*labelbox.schema.project.Project method*), 32
create_project() (*labelbox.client.Client method*), 1
create_review() (*labelbox.schema.label.Label method*), 16
created_at (*labelbox.schema.benchmark.Benchmark attribute*), 7
created_at (*labelbox.schema.bulk_import_request.BulkImportRequest attribute*), 9
created_at (*labelbox.schema.data_row.DataRow attribute*), 11
created_at (*labelbox.schema.dataset.Dataset attribute*), 13

created_at (*labelbox.schema.ontology.Ontology attribute*), 23
created_at (*labelbox.schema.organization.Organization attribute*), 25
created_at (*labelbox.schema.prediction.Prediction attribute*), 27
created_at (*labelbox.schema.prediction.PredictionModel attribute*), 29
created_at (*labelbox.schema.project.Project attribute*), 31
created_at (*labelbox.schema.review.Review attribute*), 35
created_at (*labelbox.schema.task.Task attribute*), 37
created_at (*labelbox.schema.user.User attribute*), 39
created_at (*labelbox.schema.webhook.Webhook attribute*), 41
created_by (*labelbox.schema.benchmark.Benchmark attribute*), 7
created_by (*labelbox.schema.bulk_import_request.BulkImportRequest attribute*), 9
created_by (*labelbox.schema.data_row.DataRow attribute*), 11
created_by (*labelbox.schema.dataset.Dataset attribute*), 13
created_by (*labelbox.schema.label.Label attribute*), 15
created_by (*labelbox.schema.ontology.Ontology attribute*), 23
created_by (*labelbox.schema.prediction.PredictionModel attribute*), 29
created_by (*labelbox.schema.project.Project attribute*), 31
created_by (*labelbox.schema.review.Review attribute*), 35
created_by (*labelbox.schema.task.Task attribute*), 37
created_tasks (*labelbox.schema.user.User attribute*), 39
customization_options (*labelbox.schema.labeling_frontend.LabelingFrontendOptions attribute*), 19

D

data_row (*labelbox.schema.label.Label attribute*), 15
data_row (*labelbox.schema.prediction.Prediction attribute*), 27
data_row_for_external_id() (*labelbox.schema.dataset.Dataset method*), 14
data_rows (*labelbox.schema.dataset.Dataset attribute*), 13
DataRow (*class in labelbox.schema.data_row*), 11
Dataset (*class in labelbox.schema.dataset*), 13
dataset (*labelbox.schema.data_row.DataRow attribute*), 11
datasets (*labelbox.schema.project.Project attribute*), 31
description (*labelbox.schema.dataset.Dataset attribute*), 13
description (*labelbox.schema.labeling_frontend.LabelingFrontend attribute*), 17
description (*labelbox.schema.ontology.Ontology attribute*), 23
description (*labelbox.schema.project.Project attribute*), 31

E

email (*labelbox.schema.user.User attribute*), 39
enable_model_assisted_labeling() (*labelbox.schema.project.Project method*), 33
error_file_url (*labelbox.schema.bulk_import_request.BulkImportRequest attribute*), 9
execute() (*labelbox.client.Client method*), 2
export_labels() (*labelbox.schema.project.Project method*), 33
extend_reservations() (*labelbox.schema.project.Project method*), 33
external_id (*labelbox.schema.data_row.DataRow attribute*), 11

G

get_dataset() (*labelbox.client.Client method*), 2
get_datasets() (*labelbox.client.Client method*), 2
get_labeling_frontends() (*labelbox.client.Client method*), 3
get_organization() (*labelbox.client.Client method*), 3
get_project() (*labelbox.client.Client method*), 3
get_projects() (*labelbox.client.Client method*), 3
get_user() (*labelbox.client.Client method*), 3

I

iframe_url_path (*labelbox.schema.labeling_frontend.LabelingFrontend attribute*), 17
input_file_url (*labelbox.schema.bulk_import_request.BulkImportRequest attribute*), 9
intercom_hash (*labelbox.schema.user.User attribute*), 39
InternalServerError, 43
InvalidAttributeError, 43
InvalidQueryError, 43
is_benchmark_reference (*labelbox.schema.label.Label attribute*), 15
is_external_viewer (*labelbox.schema.user.User attribute*), 39

is_viewer (*labelbox.schema.user.User attribute*), 39

L

Label (*class in labelbox.schema.label*), 15
label (*labelbox.schema.label.Label attribute*), 15
label (*labelbox.schema.prediction.Prediction attribute*), 27
label (*labelbox.schema.review.Review attribute*), 35
labelbox.client
    module, 1
labelbox.exceptions
    module, 43
labelbox.pagination
    module, 45
labelbox.schema.asset_metadata
    module, 5
labelbox.schema.benchmark
    module, 7
labelbox.schema.bulk_import_request
    module, 9
labelbox.schema.data_row
    module, 11
labelbox.schema.dataset
    module, 13
labelbox.schema.enums
    module, 47
labelbox.schema.label
    module, 15
labelbox.schema.labeling_frontend
    module, 17
labelbox.schema.ontology
    module, 23
labelbox.schema.organization
    module, 25
labelbox.schema.prediction
    module, 27
labelbox.schema.project
    module, 31
labelbox.schema.review
    module, 35
labelbox.schema.task
    module, 37
labelbox.schema.user
    module, 39
labelbox.schema.webhook
    module, 41
LabelboxError, 43
labeler_performance() (*labelbox.schema.project.Project method*), 33
labeling_frontend (*labelbox.schema.labeling_frontend.LabelingFrontendOptions attribute*), 19
labeling_frontend (*labelbox.schema.project.Project attribute*), 32
labeling_frontend_options (*labelbox.schema.project.Project attribute*), 32
labeling_parameter_overrides (*labelbox.schema.project.Project attribute*), 32
LabelingFrontend (*class in labelbox.schema.labeling_frontend*), 17
labels (*labelbox.schema.data_row.DataRow attribute*), 11
labels() (*labelbox.schema.project.Project method*), 33
last_activity (*labelbox.schema.benchmark.Benchmark attribute*), 7
last_activity_time (*labelbox.schema.project.Project attribute*), 31

M

MalformedQueryException, 43
meta_type (*labelbox.schema.asset_metadata.AssetMetadata attribute*), 5
meta_value (*labelbox.schema.asset_metadata.AssetMetadata attribute*), 5
metadata (*labelbox.schema.data_row.DataRow attribute*), 11
module
    labelbox.client, 1
    labelbox.exceptions, 43
    labelbox.pagination, 45
    labelbox.schema.asset_metadata, 5
    labelbox.schema.benchmark, 7
    labelbox.schema.bulk_import_request, 9
    labelbox.schema.data_row, 11
    labelbox.schema.dataset, 13
    labelbox.schema.enums, 47
    labelbox.schema.label, 15
    labelbox.schema.labeling_frontend, 17
    labelbox.schema.ontology, 23
    labelbox.schema.organization, 25
    labelbox.schema.prediction, 27
    labelbox.schema.project, 31
    labelbox.schema.review, 35
    labelbox.schema.task, 37
    labelbox.schema.user, 39
    labelbox.schema.webhook, 41

N

name (*labelbox.schema.bulk_import_request.BulkImportRequest attribute*), 9
name (*labelbox.schema.dataset.Dataset attribute*), 13
name (*labelbox.schema.labeling_frontend.LabelingFrontend attribute*), 17
name (*labelbox.schema.ontology.Ontology attribute*), 23

name (*labelbox.schema.organization.Organization attribute*), 25
name (*labelbox.schema.prediction.PredictionModel attribute*), 29
name (*labelbox.schema.project.Project attribute*), 31
name (*labelbox.schema.task.Task attribute*), 37
name (*labelbox.schema.user.User attribute*), 39
NetworkError, 43
nickname (*labelbox.schema.user.User attribute*), 39
normalized (*labelbox.schema.ontology.Ontology attribute*), 23
number_of_labels (*labelbox.schema.project.LabelingParameterOverride attribute*), 21

O

object_schema_count (*labelbox.schema.ontology.Ontology attribute*), 23
Ontology (*class in labelbox.schema.ontology*), 23
ontology (*labelbox.schema.project.Project attribute*), 32
Organization (*class in labelbox.schema.organization*), 25
organization (*labelbox.schema.data_row.DataRow attribute*), 11
organization (*labelbox.schema.dataset.Dataset attribute*), 13
organization (*labelbox.schema.labeling_frontend.LabelingFrontendOptions attribute*), 19
organization (*labelbox.schema.prediction.Prediction attribute*), 27
organization (*labelbox.schema.prediction.PredictionModel attribute*), 29
organization (*labelbox.schema.project.Project attribute*), 31
organization (*labelbox.schema.review.Review attribute*), 35
organization (*labelbox.schema.task.Task attribute*), 37
organization (*labelbox.schema.user.User attribute*), 39

P

PaginatedCollection (*class in labelbox.pagination*), 45
picture (*labelbox.schema.user.User attribute*), 39
Prediction (*class in labelbox.schema.prediction*), 27
prediction_model (*labelbox.schema.prediction.Prediction attribute*), 27
predictions (*labelbox.schema.data_row.DataRow attribute*), 11
predictions (*labelbox.schema.project.Project attribute*), 32
priority (*labelbox.schema.project.LabelingParameterOverride attribute*), 21
Project (*class in labelbox.schema.project*), 31
project (*labelbox.schema.bulk_import_request.BulkImportRequest attribute*), 9
project (*labelbox.schema.label.Label attribute*), 15
project (*labelbox.schema.labeling_frontend.LabelingFrontendOptions attribute*), 19
project (*labelbox.schema.prediction.Prediction attribute*), 27
project (*labelbox.schema.review.Review attribute*), 35
projects (*labelbox.schema.dataset.Dataset attribute*), 13
projects (*labelbox.schema.labeling_frontend.LabelingFrontend attribute*), 17
projects (*labelbox.schema.ontology.Ontology attribute*), 23
projects (*labelbox.schema.organization.Organization attribute*), 25
projects (*labelbox.schema.user.User attribute*), 39

R

reference_label (*labelbox.schema.benchmark.Benchmark attribute*), 7
refresh() (*labelbox.schema.bulk_import_request.BulkImportRequest method*), 9
refresh() (*labelbox.schema.task.Task method*), 37
ResourceNotFoundError, 43
Review (*class in labelbox.schema.review*), 35
Review.NetScore (*class in labelbox.schema.review*), 35
review_metrics() (*labelbox.schema.project.Project method*), 33
reviews (*labelbox.schema.label.Label attribute*), 15
reviews (*labelbox.schema.project.Project attribute*), 31
row_data (*labelbox.schema.data_row.DataRow attribute*), 11

S

score (*labelbox.schema.review.Review attribute*), 35
seconds_to_label (*labelbox.schema.label.Label attribute*), 15
set_labeling_parameter_overrides() (*labelbox.schema.project.Project method*), 33
setup() (*labelbox.schema.project.Project method*), 34
setup_complete (*labelbox.schema.project.Project attribute*), 31

slug (*labelbox.schema.prediction.PredictionModel attribute*), 29
state (*labelbox.schema.bulk_import_request.BulkImportRequest attribute*), 9
status (*labelbox.schema.task.Task attribute*), 37
status (*labelbox.schema.webhook.Webhook attribute*), 41
status_file_url (*labelbox.schema.bulk_import_request.BulkImportRequest attribute*), 9

T

Task (*class in labelbox.schema.task*), 37
TimeoutError, 44
tools() (*labelbox.schema.ontology.Ontology method*), 23
topics (*labelbox.schema.webhook.Webhook attribute*), 41

U

unset_labeling_parameter_overrides() (*labelbox.schema.project.Project method*), 34
update() (*labelbox.schema.webhook.Webhook method*), 41
updated_at (*labelbox.schema.data_row.DataRow attribute*), 11
updated_at (*labelbox.schema.dataset.Dataset attribute*), 13
updated_at (*labelbox.schema.ontology.Ontology attribute*), 23
updated_at (*labelbox.schema.organization.Organization attribute*), 25
updated_at (*labelbox.schema.prediction.Prediction attribute*), 27
updated_at (*labelbox.schema.prediction.PredictionModel attribute*), 29
updated_at (*labelbox.schema.project.Project attribute*), 31
updated_at (*labelbox.schema.review.Review attribute*), 35
updated_at (*labelbox.schema.task.Task attribute*), 37
updated_at (*labelbox.schema.user.User attribute*), 39
updated_at (*labelbox.schema.webhook.Webhook attribute*), 41
upload_annotations() (*labelbox.schema.project.Project method*), 34
upsert_review_queue() (*labelbox.schema.project.Project method*), 34
url (*labelbox.schema.webhook.Webhook attribute*), 41
User (*class in labelbox.schema.user*), 39
users (*labelbox.schema.organization.Organization attribute*), 25
UuidError, 44

V

ValidationFailedError, 44
version (*labelbox.schema.prediction.PredictionModel attribute*), 29

W

wait_till_done() (*labelbox.schema.task.Task method*), 37
wait_until_done() (*labelbox.schema.bulk_import_request.BulkImportRequest method*), 9
Webhook (*class in labelbox.schema.webhook*), 41
webhooks (*labelbox.schema.organization.Organization attribute*), 25
webhooks (*labelbox.schema.project.Project attribute*), 32