

---

# Labelbox Python API reference

*Release 2.4*

unknown

Jun 11, 2021



**CONTENTS:**

<b>1</b>	<b>Client</b>	<b>1</b>
<b>2</b>	<b>AssetMetadata</b>	<b>5</b>
<b>3</b>	<b>Benchmark</b>	<b>7</b>
<b>4</b>	<b>BulkImportRequest</b>	<b>9</b>
<b>5</b>	<b>DataRow</b>	<b>11</b>
<b>6</b>	<b>Dataset</b>	<b>13</b>
<b>7</b>	<b>Label</b>	<b>15</b>
<b>8</b>	<b>LabelingFrontend</b>	<b>17</b>
<b>9</b>	<b>LabelingFrontendOptions</b>	<b>19</b>
<b>10</b>	<b>LabelingParameterOverride</b>	<b>21</b>
<b>11</b>	<b>Ontology</b>	<b>23</b>
<b>12</b>	<b>Organization</b>	<b>25</b>
<b>13</b>	<b>Prediction</b>	<b>27</b>
<b>14</b>	<b>PredictionModel</b>	<b>29</b>
<b>15</b>	<b>Project</b>	<b>31</b>
<b>16</b>	<b>Review</b>	<b>37</b>
<b>17</b>	<b>Task</b>	<b>39</b>
<b>18</b>	<b>User</b>	<b>41</b>
<b>19</b>	<b>Webhook</b>	<b>43</b>
<b>20</b>	<b>Exceptions</b>	<b>45</b>
<b>21</b>	<b>Pagination</b>	<b>47</b>
<b>22</b>	<b>Enums</b>	<b>49</b>

<b>Python Module Index</b>	<b>51</b>
<b>Index</b>	<b>53</b>

## 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=30.0*)

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()
```





## **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`

**class** `MetaType` (*value*)

Bases: `enum.Enum`

An enumeration.



## 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



## 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

`labelbox.schema.bulk_import_request.get_mal_schemas(ontology)`

Converts a project ontology to a dict for easier lookup during ndjson validation

**Parameters** `ontology` (*Ontology*) –

**Returns** Useful for looking up a tool from a given feature schema id

**Return type** Dict

`labelbox.schema.bulk_import_request.parse_classification(tool)`

Parses a classification from an ontology. Only radio, checklist, and text are supported for mal

**Parameters** `tool` (*dict*) –

**Returns** dict

## 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, IMAGE\_OVERLAY (AssetMetadata.MetaType)
- **meta\_value** (*str*) – Asset metadata value.

**Returns** *AssetMetadata* DB object.

**Raises** **ValueError** – meta\_type must be one of the supported types.



## 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 DataRow.

**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.

**`data_rows_for_external_id`** (*external\_id*, *limit=10*)

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`.
- **`limit`** (*int*) – The maximum number of data rows to return for the given *external\_id*

**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.

## 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 (anno-
    tations).

    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.

## 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`



## **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
```





## **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



## 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. ..
    attribute:: 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.
```

```
class labelbox.schema.ontology.OntologyBuilder (tools: List[labelbox.schema.ontology.Tool]
                                              = <factory>,      classifications:
                                              List[labelbox.schema.ontology.Classification]
                                              = <factory>)
```

Bases: object

A class to help create an ontology for a Project. This should be used for making Project ontologies from scratch. OntologyBuilder can also pull from an already existing Project's ontology.

There are no required instantiation arguments.

To create an ontology, use the asdict() method after fully building your ontology within this class, and inserting it into project.setup() as the "labeling\_frontend\_options" parameter.

### Example

```
builder = OntologyBuilder() ... frontend = list(client.get_labeling_frontends())[0] project.setup(frontend,
builder.asdict())
```

```
tools
(list)
```

```
classifications
(list)
```

## 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



## 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`





## 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
```



## 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.exceptions.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. :param queue\_type: Either “LabelingQueue” or “ReviewQueue” :type queue\_type: str

**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.

**See information on priority here:** <https://docs.labelbox.com/en/configure-editor/queue-system#reservation-system>

```
>>> 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<int>, number\_of\_labels<int>) for the new override.

**Priority:**

- **Data will be labeled in priority order.**
  - A lower number priority is labeled first.
  - Minimum priority is 1.
- **Priority is not the queue position.**
  - The position is determined by the relative priority.
  - E.g. [(data\_row\_1, 5,1), (data\_row\_2, 2,1), (data\_row\_3, 10,1)] will be assigned in the following order: [data\_row\_2, data\_row\_1, data\_row\_3]
- Datarows with parameter overrides will appear before datarows without overrides.
- **The priority only effects items in the queue.**
  - Assigning a priority will not automatically add the item back into the queue.

**Number of labels:**

- **The number of times a data row should be labeled.**
  - Creates duplicate data rows in a project (one for each number of labels).
- **New duplicated data rows will be added to the queue.**
  - Already labeled duplicates will not be sent back to the queue.
- **The queue will never assign the same datarow to a single labeler more than once.**
  - If the number of labels is greater than the number of labelers working on a project then the extra items will remain in the queue (this can be fixed by removing the override at any time).
- Setting this to 1 will result in the default behavior (no duplicates).

**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.

- This will remove unlabeled duplicates in the queue.

**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]]*, *validate: bool = True*) → *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
- **validate** (*bool*) – Whether or not to validate the payload before uploading.

**Returns** BulkImportRequest

**upsert\_instructions** (*instructions\_file: str*)

- Uploads instructions to the UI. Running more than once will replace the instructions

**Parameters** **instructions\_file** (*str*) – Path to a local file. \* Must be either a pdf, text, or html file.

**Raises** **ValueError** –

- project must be setup \* instructions file must end with one of “.text”, “.txt”, “.pdf”, “.html”

**upsert\_review\_queue** (*quota\_factor*)

Sets the the proportion of total assets in a project to review.

**More information can be found here:** <https://docs.labelbox.com/en/quality-assurance/review-labels#configure-review-percentage>

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





## 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.
```



## 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.
```



## 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
```



## 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 REVIEW_CREATED, RE-
        VIEW_UPDATED, REVIEW_DELETED
        Type str

    status
        ACTIVE, INACTIVE, REVOKED
        Type str

class Status (value)
    Bases: enum.Enum

    An enumeration.

class Topic (value)
    Bases: enum.Enum

    An enumeration.

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 notifica-
          tions for. Must be one of Webhook.Topic
        • 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.

**Raises** **ValueError** – If the topic is not one of Topic or status is not one of Status

**Information on configuring your server can be found here (this is where the url points to and the secret is set).**

<https://docs.labelbox.com/en/configure-editor/webhooks-setup#setup-steps>

**delete()**

Deletes the webhook

**update** (*topics=None, url=None, status=None*)

Updates the Webhook.

**Parameters**

- **topics** (*Optional[List[Topic]]*) – The new topics.
- **Optional[str]** (*url*) – The new URL value.
- **status** (*Optional[Status]*) – The new status. If an argument is set to None then no updates will be made to that field.



## 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.InconsistentOntologyException`  
Bases: `Exception`

**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.MALValidationError` (*message, cause=None*)  
Bases: `labelbox.exceptions.LabelboxError`

Raised when user input is invalid for MAL imports.

**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.

## 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.



## 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

### I

- `labelbox.client`, 1
- `labelbox.exceptions`, 45
- `labelbox.pagination`, 47
- `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`, 49
- `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`, 37
- `labelbox.schema.task`, 39
- `labelbox.schema.user`, 41
- `labelbox.schema.webhook`, 43





## Symbols

`__init__()` (*labelbox.client.Client* method), 1  
`__init__()` (*labelbox.pagination.PaginatedCollection* method), 47

## 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`, 45  
`AssetMetadata` (class in *labelbox.schema.asset\_metadata*), 5  
`AssetMetadata.MetaType` (class in *labelbox.schema.asset\_metadata*), 5  
`AuthenticationError`, 45  
`AuthorizationError`, 45  
`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*), 49

## C

`classification_schema_count` (*labelbox.schema.ontology.Ontology* attribute), 23  
`classifications` (*labelbox.schema.ontology.OntologyBuilder* attribute), 24  
`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), 39  
`create()` (*labelbox.schema.webhook.Webhook* static method), 43  
`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), 37
  - `created_at` (`labelbox.schema.task.Task` attribute), 39
  - `created_at` (`labelbox.schema.user.User` attribute), 41
  - `created_at` (`labelbox.schema.webhook.Webhook` attribute), 43
  - `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), 37
  - `created_by` (`labelbox.schema.task.Task` attribute), 39
  - `created_tasks` (`labelbox.schema.user.User` attribute), 41
  - `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
  - `data_rows_for_external_id()` (`labelbox.schema.dataset.Dataset` method), 14
  - `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
  - `delete()` (`labelbox.schema.webhook.Webhook` method), 44
  - `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), 41
  - `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_mal_schemas()` (in module `labelbox.schema.bulk_import_request`), 10
  - `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
  - `InconsistentOntologyException`, 45

intercom\_hash (labelbox.schema.user.User attribute), 41  
InternalServerError, 45  
InvalidAttributeError, 45  
InvalidQueryError, 45  
is\_benchmark\_reference (labelbox.schema.label.Label attribute), 15  
is\_external\_viewer (labelbox.schema.user.User attribute), 41  
is\_viewer (labelbox.schema.user.User attribute), 41  
**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), 37  
labelbox.client  
  module, 1  
labelbox.exceptions  
  module, 45  
labelbox.pagination  
  module, 47  
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, 49  
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, 37  
labelbox.schema.task  
  module, 39  
labelbox.schema.user  
  module, 41  
labelbox.schema.webhook  
  module, 43  
LabelboxError, 45  
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, 45  
MALValidationError, 45  
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, 45  
  labelbox.pagination, 47  
  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, 49  
  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, 37  
 labelbox.schema.task, 39  
 labelbox.schema.user, 41  
 labelbox.schema.webhook, 43

## 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.organization.Organization attribute), 25  
 name (labelbox.schema.prediction.PredictionModel attribute), 29  
 name (labelbox.schema.project.Project attribute), 31  
 name (labelbox.schema.task.Task attribute), 39  
 name (labelbox.schema.user.User attribute), 41  
 NetworkError, 46  
 nickname (labelbox.schema.user.User attribute), 41  
 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  
 OntologyBuilder (class in labelbox.schema.ontology), 23  
 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), 37  
 organization (labelbox.schema.task.Task attribute), 39  
 organization (labelbox.schema.user.User attribute), 41

## P

PaginatedCollection (class in labelbox.pagination), 47  
 parse\_classification() (in module labelbox.schema.bulk\_import\_request), 10  
 picture (labelbox.schema.user.User attribute), 41  
 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), 37  
 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), 41

## R

reference\_label (labelbox.schema.benchmark.Benchmark attribute), 7  
 refresh() (labelbox.schema.bulk\_import\_request.BulkImportRequest method), 9  
 refresh() (labelbox.schema.task.Task method), 39  
 ResourceNotFoundError, 46  
 Review (class in labelbox.schema.review), 37  
 Review.NetScore (class in labelbox.schema.review), 37

[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), 37  
[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), 39  
[status](#) ([labelbox.schema.webhook.Webhook](#) attribute), 43  
[status\\_file\\_url](#) ([labelbox.schema.bulk\\_import\\_request.BulkImportRequest](#) attribute), 9

## T

[Task](#) (class in [labelbox.schema.task](#)), 39  
[TimeoutError](#), 46  
[tools](#) ([labelbox.schema.ontology.OntologyBuilder](#) attribute), 24  
[tools\(\)](#) ([labelbox.schema.ontology.Ontology](#) method), 23  
[topics](#) ([labelbox.schema.webhook.Webhook](#) attribute), 43

## U

[unset\\_labeling\\_parameter\\_overrides\(\)](#) ([labelbox.schema.project.Project](#) method), 34  
[update\(\)](#) ([labelbox.schema.webhook.Webhook](#) method), 44  
[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.PredictionModel](#) 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), 37  
[updated\\_at](#) ([labelbox.schema.task.Task](#) attribute), 39  
[updated\\_at](#) ([labelbox.schema.user.User](#) attribute), 41  
[updated\\_at](#) ([labelbox.schema.webhook.Webhook](#) attribute), 43  
[upload\\_annotations\(\)](#) ([labelbox.schema.project.Project](#) method), 34  
[upsert\\_instructions\(\)](#) ([labelbox.schema.project.Project](#) method), 35  
[upsert\\_review\\_queue\(\)](#) ([labelbox.schema.project.Project](#) method), 35  
[url](#) ([labelbox.schema.webhook.Webhook](#) attribute), 43  
[User](#) (class in [labelbox.schema.user](#)), 41  
[users](#) ([labelbox.schema.organization.Organization](#) attribute), 25  
[UuidError](#), 46

## V

[ValidationFailedError](#), 46  
[version](#) ([labelbox.schema.prediction.PredictionModel](#) attribute), 29

## W

[wait\\_till\\_done\(\)](#) ([labelbox.schema.task.Task](#) method), 39  
[wait\\_until\\_done\(\)](#) ([labelbox.schema.bulk\\_import\\_request.BulkImportRequest](#) method), 9  
[Webhook](#) (class in [labelbox.schema.webhook](#)), 43  
[Webhook.Status](#) (class in [labelbox.schema.webhook](#)), 43  
[Webhook.Topic](#) (class in [labelbox.schema.webhook](#)), 43  
[webhooks](#) ([labelbox.schema.organization.Organization](#) attribute), 25  
[webhooks](#) ([labelbox.schema.project.Project](#) attribute), 32