

---

# **Feast Documentation**

**Feast Authors**

**Mar 21, 2021**



# CONTENTS

|          |                            |           |
|----------|----------------------------|-----------|
| <b>1</b> | <b>Client</b>              | <b>1</b>  |
| <b>2</b> | <b>Data Source</b>         | <b>7</b>  |
| <b>3</b> | <b>Entity</b>              | <b>11</b> |
| <b>4</b> | <b>Feature Table</b>       | <b>13</b> |
| <b>5</b> | <b>Feature</b>             | <b>15</b> |
| <b>6</b> | <b>Constants</b>           | <b>17</b> |
|          | <b>Python Module Index</b> | <b>21</b> |
|          | <b>Index</b>               | <b>23</b> |



## CLIENT

**class** `feast.client.Client` (*options: Optional[Dict[str, str]] = None, \*\*kwargs*)

Feast Client: Used for creating, managing, and retrieving features.

**apply** (*objects: Union[List[Union[feast.entity.Entity, feast.feature\_table.FeatureTable]], feast.entity.Entity, feast.feature\_table.FeatureTable], project: Optional[str] = None*)

Idempotently registers entities and feature tables with Feast Core. Either a single entity or feature table or a list can be provided.

**Parameters** `objects` – List of entities and/or feature tables that will be registered

### Examples

```
>>> from feast import Client
>>> from feast.entity import Entity
>>> from feast.value_type import ValueType
>>>
>>> feast_client = Client(core_url="localhost:6565")
>>> entity = Entity(
>>>     name="driver_entity",
>>>     description="Driver entity for car rides",
>>>     value_type=ValueType.STRING,
>>>     labels={
>>>         "key": "val"
>>>     }
>>> )
>>> feast_client.apply(entity)
```

**apply\_entity** (*entities: Union[List[feast.entity.Entity], feast.entity.Entity], project: Optional[str] = None*)

Deprecated. Please see `apply()`.

**apply\_feature\_table** (*feature\_tables: Union[List[feast.feature\_table.FeatureTable], feast.feature\_table.FeatureTable], project: Optional[str] = None*)

Deprecated. Please see `apply()`.

**archive\_project** (*project*)

Archives a project. Project will still continue to function for ingestion and retrieval, but will be in a read-only state. It will also not be visible from the Core API for management purposes.

**Parameters** `project` – Name of project to archive

**property** `core_secure`

Retrieve Feast Core client-side SSL/TLS setting

**Returns** Whether client-side SSL/TLS is enabled

**property core\_url**

Retrieve Feast Core URL

**Returns** Feast Core URL string

**create\_project** (*project: str*)

Creates a Feast project

**Parameters** **project** – Name of project

**delete\_feature\_table** (*name: str, project: Optional[str] = None*) → *None*

Deletes a feature table.

**Parameters**

- **project** – Feast project that this feature table belongs to
- **name** – Name of feature table

**get\_entity** (*name: str, project: Optional[str] = None*) → *feast.entity.Entity*

Retrieves an entity.

**Parameters**

- **project** – Feast project that this entity belongs to
- **name** – Name of entity

**Returns** Returns either the specified entity, or raises an exception if none is found

**get\_feature\_table** (*name: str, project: Optional[str] = None*) → *feast.feature\_table.FeatureTable*

Retrieves a feature table.

**Parameters**

- **project** – Feast project that this feature table belongs to
- **name** – Name of feature table

**Returns** Returns either the specified feature table, or raises an exception if none is found

**get\_online\_features** (*feature\_refs: List[str], entity\_rows: List[Dict[str, Any]], project: Optional[str] = None*) → *feast.online\_response.OnlineResponse*

Retrieves the latest online feature data from Feast Serving. :param feature\_refs: List of feature references that will be returned for each entity.

Each feature reference should have the following format: “feature\_table:feature” where “feature\_table” & “feature” refer to the feature and feature table names respectively. Only the feature name is required.

**Parameters**

- **entity\_rows** – A list of dictionaries where each key-value is an entity-name, entity-value pair.
- **project** – Optionally specify the the project override. If specified, uses given project for retrieval. Overrides the projects specified in Feature References if also are specified.

**Returns** GetOnlineFeaturesResponse containing the feature data in records. Each EntityRow provided will yield one record, which contains data fields with data value and field status metadata (if included).

## Examples

```
>>> from feast import Client
>>>
>>> feast_client = Client(core_url="localhost:6565", serving_url=
↳ "localhost:6566")
>>> feature_refs = ["sales:daily_transactions"]
>>> entity_rows = [{"customer_id": 0}, {"customer_id": 1}]
>>>
>>> online_response = feast_client.get_online_features(
>>>     feature_refs, entity_rows, project="my_project")
>>> online_response_dict = online_response.to_dict()
>>> print(online_response_dict)
{'sales:daily_transactions': [1.1, 1.2], 'sales:customer_id': [0, 1]}
```

**ingest** (*feature\_table*: *Union[str, feast.feature\_table.FeatureTable]*, *source*: *Union[pandas.core.frame.DataFrame, str]*, *project*: *Optional[str]* = *None*, *chunk\_size*: *int* = 10000, *max\_workers*: *int* = 1, *timeout*: *int* = 120) → *None*  
Batch load feature data into a FeatureTable.

### Parameters

- **feature\_table** (*typing.Union[str, feast.feature\_table.FeatureTable]*) – FeatureTable object or the string name of the feature table
- **source** (*typing.Union[pd.DataFrame, str]*) – Either a file path or Pandas Dataframe to ingest into Feast Files that are currently supported:
  - parquet
  - csv
  - json
- **project** – Feast project to locate FeatureTable
- **chunk\_size** (*int*) – Amount of rows to load and ingest at a time.
- **max\_workers** (*int*) – Number of worker processes to use to encode values.
- **timeout** (*int*) – Timeout in seconds to wait for completion.

## Examples

```
>>> from feast import Client
>>>
>>> client = Client(core_url="localhost:6565")
>>> ft_df = pd.DataFrame(
>>>     {
>>>         "datetime": [pd.datetime.now()],
>>>         "driver": [1001],
>>>         "rating": [4.3],
>>>     }
>>> )
>>> client.set_project("project1")
>>>
>>> driver_ft = client.get_feature_table("driver")
>>> client.ingest(driver_ft, ft_df)
```

**property job\_service\_secure**

Retrieve Feast Job Service client-side SSL/TLS setting

**Returns** Whether client-side SSL/TLS is enabled**property job\_service\_url**

Retrieve Feast Job Service URL

**Returns** Feast Job Service URL string**list\_entities** (*project: Optional[str] = None, labels: Dict[str, str] = {}*) → List[*feast.entity.Entity*]

Retrieve a list of entities from Feast Core

**Parameters**

- **project** – Filter entities based on project name
- **labels** – User-defined labels that these entities are associated with

**Returns** List of entities**list\_feature\_tables** (*project: Optional[str] = None, labels: Dict[str, str] = {}*) → List[*feast.feature\_table.FeatureTable*]

Retrieve a list of feature tables from Feast Core

**Parameters** **project** – Filter feature tables based on project name**Returns** List of feature tables**list\_features\_by\_ref** (*project: Optional[str] = None, entities: List[str] = [], labels: Dict[str, str] = {}*) → Dict[*feast.feature.FeatureRef*, *feast.feature.Feature*]

Retrieve a dictionary of feature reference to feature from Feast Core based on filters provided.

**Parameters**

- **project** – Feast project that these features belongs to
- **entities** – Feast entity that these features are associated with
- **labels** – Feast labels that these features are associated with

**Returns** features>**Return type** Dictionary of <feature references**Examples**

```
>>> from feast import Client
>>>
>>> feast_client = Client(core_url="localhost:6565")
>>> features = feast_client.list_features(project="test_project", entities=[
↳ "driver_id", labels={"key1": "val1", "key2": "val2"}])
>>> print(features)
```

**list\_projects** () → List[str]

List all active Feast projects

**Returns** List of project names**property project**

Retrieve currently active project

**Returns** Project name



**property serving\_secure**

Retrieve Feast Serving client-side SSL/TLS setting

**Returns** Whether client-side SSL/TLS is enabled

**property serving\_url**

Retrieve Feast Serving URL

**Returns** Feast Serving URL string

**set\_project** (*project: Optional[str] = None*)

Set currently active Feast project

**Parameters** **project** – Project to set as active. If unset, will reset to the default project.

**version** (*sdk\_only=False*)

Returns version information from Feast Core and Feast Serving



## DATA SOURCE

```

class feast.data_source.BigQueryOptions (table_ref: Optional[str], query: Optional[str])
    DataSource BigQuery options used to source features from BigQuery query

classmethod from_proto (bigquery_options_proto: feast.core.DataSource_pb2.BigQueryOptions)
    Creates a BigQueryOptions from a protobuf representation of a BigQuery option

    Parameters bigquery_options_proto – A protobuf representation of a DataSource

    Returns Returns a BigQueryOptions object based on the bigquery_options protobuf

property query
    Returns the BigQuery SQL query referenced by this source

property table_ref
    Returns the table ref of this BQ table

to_proto () → feast.core.DataSource_pb2.BigQueryOptions
    Converts an BigQueryOptionsProto object to its protobuf representation.

    Returns BigQueryOptionsProto protobuf

class feast.data_source.BigQuerySource (event_timestamp_column: str, table_ref: Optional[str] = None, created_timestamp_column: Optional[str] = "", field_mapping: Optional[Dict[str, str]] = None, date_partition_column: Optional[str] = "", query: Optional[str] = None)

    property bigquery_options
    Returns the bigquery options of this data source

    get_table_query_string () → str
    Returns a string that can directly be used to reference this table in SQL

    to_proto () → feast.core.DataSource_pb2.DataSource
    Converts an DataSourceProto object to its protobuf representation.

class feast.data_source.DataSource (event_timestamp_column: str, created_timestamp_column: Optional[str] = "", field_mapping: Optional[Dict[str, str]] = None, date_partition_column: Optional[str] = "")
    DataSource that can be used source features

    property created_timestamp_column
    Returns the created timestamp column of this data source

    property date_partition_column
    Returns the date partition column of this data source

```

**property event\_timestamp\_column**  
Returns the event timestamp column of this data source

**property field\_mapping**  
Returns the field mapping of this data source

**static from\_proto** (*data\_source*)  
Convert data source config in FeatureTable spec to a DataSource class object.

**to\_proto** () → `feast.core.DataSource_pb2.DataSource`  
Converts an DataSourceProto object to its protobuf representation.

**class** `feast.data_source.FileOptions` (*file\_format*: *Optional[feast.data\_format.FileFormat]*,  
*file\_url*: *Optional[str]*)  
DataSource File options used to source features from a file

**property file\_format**  
Returns the file format of this file

**property file\_url**  
Returns the file url of this file

**classmethod from\_proto** (*file\_options\_proto*: *feast.core.DataSource\_pb2.FileOptions*)  
Creates a FileOptions from a protobuf representation of a file option

**Parameters** *file\_options\_proto* – a protobuf representation of a datasource

**Returns** Returns a FileOptions object based on the file\_options protobuf

**to\_proto** () → `feast.core.DataSource_pb2.FileOptions`  
Converts an FileOptionsProto object to its protobuf representation.

**Returns** FileOptionsProto protobuf

**class** `feast.data_source.FileSource` (*event\_timestamp\_column*: *str*, *file\_url*: *Optional[str]*  
*= None*, *path*: *Optional[str]* = *None*, *file\_format*:  
*Optional[feast.data\_format.FileFormat]* = *None*,  
*created\_timestamp\_column*: *Optional[str]* = *"*,  
*field\_mapping*: *Optional[Dict[str, str]]* = *None*,  
*date\_partition\_column*: *Optional[str]* = *"*)

**property file\_options**  
Returns the file options of this data source

**property path**  
Returns the file path of this feature data source

**to\_proto** () → `feast.core.DataSource_pb2.DataSource`  
Converts an DataSourceProto object to its protobuf representation.

**class** `feast.data_source.KafkaOptions` (*bootstrap\_servers*: *str*, *message\_format*:  
*feast.data\_format.StreamFormat*, *topic*: *str*)  
DataSource Kafka options used to source features from Kafka messages

**property bootstrap\_servers**  
Returns a comma-separated list of Kafka bootstrap servers

**classmethod from\_proto** (*kafka\_options\_proto*: *feast.core.DataSource\_pb2.KafkaOptions*)  
Creates a KafkaOptions from a protobuf representation of a kafka option

**Parameters** *kafka\_options\_proto* – A protobuf representation of a DataSource

**Returns** Returns a BigQueryOptions object based on the kafka\_options protobuf

**property message\_format**

Returns the data format that is used to encode the feature data in Kafka messages

**to\_proto()** → feast.core.DataSource\_pb2.KafkaOptions

Converts an KafkaOptionsProto object to its protobuf representation.

**Returns** KafkaOptionsProto protobuf

**property topic**

Returns the Kafka topic to collect feature data from

```
class feast.data_source.KafkaSource(event_timestamp_column: str, bootstrap_servers: str,
                                     message_format: feast.data_format.StreamFormat,
                                     topic: str, created_timestamp_column: Optional[str]
                                     = "", field_mapping: Optional[Dict[str, str]] = {},
                                     date_partition_column: Optional[str] = "")
```

**property kafka\_options**

Returns the kafka options of this data source

**to\_proto()** → feast.core.DataSource\_pb2.DataSource

Converts an DataSourceProto object to its protobuf representation.

```
class feast.data_source.KinesisOptions(record_format: feast.data_format.StreamFormat, re-
                                     gion: str, stream_name: str)
```

DataSource Kinesis options used to source features from Kinesis records

**classmethod from\_proto**(kinesis\_options\_proto: feast.core.DataSource\_pb2.KinesisOptions)

Creates a KinesisOptions from a protobuf representation of a kinesis option

**Parameters kinesis\_options\_proto** – A protobuf representation of a DataSource

**Returns** Returns a KinesisOptions object based on the kinesis\_options protobuf

**property record\_format**

Returns the data format used to encode the feature data in the Kinesis records.

**property region**

Returns the AWS region of Kinesis stream

**property stream\_name**

Returns the Kinesis stream name to obtain feature data from

**to\_proto()** → feast.core.DataSource\_pb2.KinesisOptions

Converts an KinesisOptionsProto object to its protobuf representation.

**Returns** KinesisOptionsProto protobuf

```
class feast.data_source.KinesisSource(event_timestamp_column: str, created_timestamp_column: str, record_format:
                                     feast.data_format.StreamFormat, region: str,
                                     stream_name: str, field_mapping: Optional[Dict[str,
                                     str]] = {}, date_partition_column: Optional[str] = "")
```

**property kinesis\_options**

Returns the kinesis options of this data source

**to\_proto()** → feast.core.DataSource\_pb2.DataSource

Converts an DataSourceProto object to its protobuf representation.

```
class feast.data_source.SourceType(value)
```

DataSource value type. Used to define source types in DataSource.



## ENTITY

**class** `feast.entity.Entity` (*name: str, description: str, value\_type: feast.value\_type.ValueType, labels: Optional[MutableMapping[str, str]] = None*)

Represents a collection of entities and associated metadata.

**property** `created_timestamp`

Returns the `created_timestamp` of this entity

**property** `description`

Returns the description of this entity

**classmethod** `from_dict` (*entity\_dict*)

Creates an entity from a dict

**Parameters** `entity_dict` – A dict representation of an entity

**Returns** Returns a `EntityV2` object based on the entity dict

**classmethod** `from_proto` (*entity\_proto: feast.core.Entity\_pb2.Entity*)

Creates an entity from a protobuf representation of an entity

**Parameters** `entity_proto` – A protobuf representation of an entity

**Returns** Returns a `EntityV2` object based on the entity protobuf

**classmethod** `from_yaml` (*yaml: str*)

Creates an entity from a YAML string body or a file path

**Parameters** `yaml` – Either a file path containing a yaml file or a YAML string

**Returns** Returns a `EntityV2` object based on the YAML file

**is\_valid** ()

Validates the state of a entity locally. Raises an exception if entity is invalid.

**property** `labels`

Returns the labels of this entity. This is the user defined metadata defined as a dictionary.

**property** `last_updated_timestamp`

Returns the `last_updated_timestamp` of this entity

**property** `name`

Returns the name of this entity

**to\_dict** () → Dict

Converts entity to dict

**Returns** Dictionary object representation of entity

**to\_proto** () → `feast.core.Entity_pb2.Entity`

Converts an entity object to its protobuf representation

**Returns** EntityV2Proto protobuf

**to\_spec\_proto** () → feast.core.Entity\_pb2.EntitySpecV2

Converts an EntityV2 object to its protobuf representation. Used when passing EntitySpecV2 object to Feast request.

**Returns** EntitySpecV2 protobuf

**to\_yaml** ()

Converts a entity to a YAML string.

**Returns** Entity string returned in YAML format

**property value\_type**

Returns the type of this entity



## FEATURE TABLE

```
class feast.feature_table.FeatureTable (name: str, entities: List[str], features:
                                         List[feast.feature.Feature], batch_source: Optional[Union[feast.data_source.BigQuerySource,
                                         feast.data_source.FileSource]]
                                         = None, stream_source: Optional[Union[feast.data_source.KafkaSource,
                                         feast.data_source.KinesisSource]]
                                         = None, max_age: Optional[google.protobuf.duration_pb2.Duration]
                                         = None, labels: Optional[MutableMapping[str,
                                         str]] = None)
```

Represents a collection of features and associated metadata.

**add\_feature** (*feature: feast.feature.Feature*)

Adds a new feature to the feature table.

**property batch\_source**

Returns the batch source of this feature table

**property created\_timestamp**

Returns the created\_timestamp of this feature table

**property entities**

Returns the entities of this feature table

**property features**

Returns the features of this feature table

**classmethod from\_dict** (*ft\_dict*)

Creates a feature table from a dict

**Parameters** *ft\_dict* – A dict representation of a feature table

**Returns** Returns a FeatureTable object based on the feature table dict

**classmethod from\_proto** (*feature\_table\_proto: feast.core.FeatureTable\_pb2.FeatureTable*)

Creates a feature table from a protobuf representation of a feature table

**Parameters** *feature\_table\_proto* – A protobuf representation of a feature table

**Returns** Returns a FeatureTableProto object based on the feature table protobuf

**classmethod from\_yaml** (*yaml: str*)

Creates a feature table from a YAML string body or a file path

**Parameters** *yaml* – Either a file path containing a yaml file or a YAML string

**Returns** Returns a FeatureTable object based on the YAML file

**is\_valid()**

Validates the state of a feature table locally. Raises an exception if feature table is invalid.

**property labels**

Returns the labels of this feature table. This is the user defined metadata defined as a dictionary.

**property last\_updated\_timestamp**

Returns the last\_updated\_timestamp of this feature table

**property max\_age**

Returns the maximum age of this feature table. This is the total maximum amount of staleness that will be allowed during feature retrieval for each specific feature that is looked up.

**property name**

Returns the name of this feature table

**property stream\_source**

Returns the stream source of this feature table

**to\_dict()** → Dict

Converts feature table to dict

**Returns** Dictionary object representation of feature table

**to\_proto()** → feast.core.FeatureTable\_pb2.FeatureTable

Converts an feature table object to its protobuf representation

**Returns** FeatureTableProto protobuf

**to\_spec\_proto()** → feast.core.FeatureTable\_pb2.FeatureTableSpec

Converts an FeatureTableProto object to its protobuf representation. Used when passing FeatureTableSpecProto object to Feast request.

**Returns** FeatureTableSpecProto protobuf

**to\_yaml()**

Converts a feature table to a YAML string.

**Returns** Feature table string returned in YAML format

## FEATURE

---

```
class feast.feature.Feature(name: str, dtype: feast.value_type.ValueType, labels: Optional[MutableMapping[str, str]] = None)
```

Feature field type

**property dtype**

Getter for data type of this field

**classmethod from\_proto** (*feature\_proto: feast.core.Feature\_pb2.FeatureSpecV2*)

**Parameters** *feature\_proto* – FeatureSpecV2 protobuf object

**Returns** Feature object

**property labels**

Getter for labels of this field

**property name**

Getter for name of this field

**to\_proto** () → *feast.core.Feature\_pb2.FeatureSpecV2*

Converts Feature object to its Protocol Buffer representation

```
class feast.feature.FeatureRef(name: str, feature_table: Optional[str] = None)
```

Feature Reference represents a reference to a specific feature.

**classmethod from\_proto** (*proto: feast.serving.ServingService\_pb2.FeatureReferenceV2*)

Construct a feature reference from the given FeatureReference proto Arg:

*proto*: Protobuf FeatureReference to construct from

**Returns** FeatureRef that refers to the given feature

**classmethod from\_str** (*feature\_ref\_str: str*)

Parse the given string feature reference into FeatureRef model String feature reference should be in the format *feature\_table:feature*. Where “*feature\_table*” and “*name*” are the *feature\_table* name and feature name respectively. :param *feature\_ref\_str*: String representation of the feature reference

**Returns** FeatureRef that refers to the given feature

**to\_proto** () → *feast.serving.ServingService\_pb2.FeatureReferenceV2*

Convert and return this feature table reference to protobuf. :returns: Protobuf representation of this feature table reference.



## CONSTANTS

```
feast.constants.CONFIG_FEAST_ENV_VAR_PREFIX: str = 'FEAST_'
    Default prefix to Feast environmental variables

feast.constants.CONFIG_FILE_DEFAULT_DIRECTORY: str = '.feast'
    Default directory to Feast configuration file

feast.constants.CONFIG_FILE_NAME: str = 'config'
    Default Feast configuration file name

feast.constants.CONFIG_FILE_SECTION: str = 'general'
    Default section in Feast configuration file to specify options

class feast.constants.ConfigOptions
    Feast Configuration Options

    AUTH_PROVIDER: str = 'auth_provider'
        Authentication Provider - Google OpenID/OAuth

        Options: "google" / "oauth"

    AUTH_TOKEN: Optional[str] = 'auth_token'
        JWT Auth token for user authentication to Feast

    AZURE_BLOB_ACCOUNT_ACCESS_KEY: Optional[str] = 'azure_blob_account_access_key'
        Account access key for Azure blob storage_client

    AZURE_BLOB_ACCOUNT_NAME: Optional[str] = 'azure_blob_account_name'
        Account name for Azure blob storage_client

    BATCH_FEATURE_REQUEST_WAIT_TIME_SECONDS: str = 'batch_feature_request_wait_time_seconds'
        Time to wait for historical feature requests before timing out.

    BATCH_INGESTION_PRODUCTION_TIMEOUT: str = 'batch_ingestion_production_timeout'
        Default timeout when running batch ingestion

    CORE_ENABLE_SSL: str = 'core_enable_ssl'
        Enable or disable TLS/SSL to Feast Core

    CORE_SERVER_SSL_CERT: str = 'core_server_ssl_cert'
        Path to certificate(s) to secure connection to Feast Core

    CORE_URL: str = 'core_url'
        Default Feast Core URL

    DATAPROC_CLUSTER_NAME: Optional[str] = 'dataproc_cluster_name'
        Dataproc cluster to run Feast Spark Jobs in

    DATAPROC_EXECUTOR_CORES = 'dataproc_executor_cores'
        No. of executor cores for Dataproc cluster
```

**DATAPROC\_EXECUTOR\_INSTANCES** = 'dataproc\_executor\_instances'

No. of executor instances for Dataproc cluster

**DATAPROC\_EXECUTOR\_MEMORY** = 'dataproc\_executor\_memory'

No. of executor memory for Dataproc cluster

**DATAPROC\_PROJECT**: Optional[str] = 'dataproc\_project'

Project of Dataproc cluster

**DATAPROC\_REGION**: Optional[str] = 'dataproc\_region'

Region of Dataproc cluster

**DEADLETTER\_PATH**: str = 'deadletter\_path'

Ingestion Job DeadLetter Destination. The choice of storage is connected to the choice of SPARK\_LAUNCHER.

Eg. gs://some-bucket/output/, s3://some-bucket/output/, file:///data/subfolder/

**EMR\_CLUSTER\_ID**: Optional[str] = 'emr\_cluster\_id'

EMR cluster to run Feast Spark Jobs in

**EMR\_CLUSTER\_TEMPLATE\_PATH**: Optional[str] = 'emr\_cluster\_template\_path'

Template path of EMR cluster

**EMR\_LOG\_LOCATION**: Optional[str] = 'emr\_log\_location'

Log path of EMR cluster

**EMR\_REGION**: Optional[str] = 'emr\_region'

Region of EMR cluster

**ENABLE\_AUTH**: str = 'enable\_auth'

Enable user authentication to Feast Core

**GRPC\_CONNECTION\_TIMEOUT**: str = 'grpc\_connection\_timeout'

Default connection timeout to Feast Serving, Feast Core, and Feast Job Service (in seconds)

**GRPC\_CONNECTION\_TIMEOUT\_APPLY**: str = 'grpc\_connection\_timeout\_apply'

Default gRPC connection timeout when sending an ApplyFeatureTable command to Feast Core (in seconds)

**HISTORICAL\_FEATURE\_OUTPUT\_FORMAT**: str = 'historical\_feature\_output\_format'

File format of historical retrieval features

**HISTORICAL\_FEATURE\_OUTPUT\_LOCATION**: Optional[str] = 'historical\_feature\_output\_location'

File location of historical retrieval features

**INGESTION\_DROP\_INVALID\_ROWS** = 'ingestion\_drop\_invalid\_rows'

If set to true rows that do not pass custom validation (see feast.contrib.validation) won't be saved to Online Storage

**JOB\_SERVICE\_ENABLE\_CONTROL\_LOOP**: str = 'job\_service\_enable\_control\_loop'

Enable or disable control loop for Feast Job Service

**JOB\_SERVICE\_ENABLE\_SSL**: str = 'job\_service\_enable\_ssl'

Enable or disable TLS/SSL to Feast Job Service

**JOB\_SERVICE\_SERVER\_SSL\_CERT**: str = 'job\_service\_server\_ssl\_cert'

Path to certificate(s) to secure connection to Feast Job Service

**JOB\_SERVICE\_URL**: Optional[str] = 'job\_service\_url'

Default Feast Job Service URL

**OAUTH\_AUDIENCE**: Optional[str] = 'oauth\_audience'

Oauth intended recipients

**OAUTH\_CLIENT\_ID:** `Optional[str] = 'oauth_client_id'`  
 OAuth client ID

**OAUTH\_CLIENT\_SECRET:** `Optional[str] = 'oauth_client_secret'`  
 OAuth client secret

**OAUTH\_GRANT\_TYPE:** `Optional[str] = 'oauth_grant_type'`  
 OAuth grant type

**OAUTH\_TOKEN\_REQUEST\_URL:** `Optional[str] = 'oauth_token_request_url'`  
 OAuth token request url

**PROJECT:** `str = 'project'`  
 Feast project namespace to use

**REDIS\_HOST:** `str = 'redis_host'`  
 Default Redis host

**REDIS\_PORT:** `str = 'redis_port'`  
 Default Redis port

**REDIS\_SSL:** `str = 'redis_ssl'`  
 Enable or disable TLS/SSL to Redis

**REGISTRY\_PATH:** `Optional[str] = 'registry_path'`  
 Object store registry

**S3\_ENDPOINT\_URL:** `Optional[str] = 's3_endpoint_url'`  
 Endpoint URL for S3 storage\_client

**SERVING\_ENABLE\_SSL:** `str = 'serving_enable_ssl'`  
 Enable or disable TLS/SSL to Feast Serving

**SERVING\_SERVER\_SSL\_CERT:** `str = 'serving_server_ssl_cert'`  
 Path to certificate(s) to secure connection to Feast Serving

**SERVING\_URL:** `str = 'serving_url'`  
 Default Feast Serving URL

**SPARK\_BQ\_MATERIALIZATION\_DATASET:** `Optional[str] = 'spark_bq_materialization_dataset'`  
 The dataset id where the materialized view of BigQuerySource is going to be created by default, use the same dataset where view is located

**SPARK\_BQ\_MATERIALIZATION\_PROJECT:** `Optional[str] = 'spark_bq_materialization_project'`  
 The project id where the materialized view of BigQuerySource is going to be created by default, use the same project where view is located

**SPARK\_HOME:** `Optional[str] = 'spark_home'`  
 Directory where Spark is installed

**SPARK\_INGESTION\_JAR:** `str = 'spark_ingestion_jar'`  
 Feast Spark Job ingestion jar file. The choice of storage is connected to the choice of SPARK\_LAUNCHER.  
  
 Eg. “dataproc” (http and gs), “emr” (http and s3), “standalone” (http and file)

**SPARK\_LAUNCHER:** `Optional[str] = 'spark_launcher'`  
 Spark Job launcher. The choice of storage is connected to the choice of SPARK\_LAUNCHER.  
  
 Options: “standalone”, “dataproc”, “emr”

**SPARK\_STAGING\_LOCATION:** `Optional[str] = 'spark_staging_location'`  
 Feast Spark Job ingestion jobs staging location. The choice of storage is connected to the choice of SPARK\_LAUNCHER.

Eg. gs://some-bucket/output/, s3://some-bucket/output/, file:///data/subfolder/

**SPARK\_STANDALONE\_MASTER:** `str` = `'spark_standalone_master'`

Spark resource manager master url

**STATSD\_ENABLED:** `str` = `'statsd_enabled'`

Enable or disable StatsD

**STATSD\_HOST:** `Optional[str]` = `'statsd_host'`

Default StatsD port

**STATSD\_PORT:** `Optional[str]` = `'statsd_port'`

Default StatsD port

**STENCIL\_URL:** `str` = `'stencil_url'`

ProtoRegistry Address (currently only Stencil Server is supported as registry) <https://github.com/gojekfarm/stencil>

**TELEMETRY** = `'telemetry'`

Telemetry enabled

`feast.constants.DATETIME_COLUMN:` `str` = `'datetime'`

Default datetime column name for point-in-time join

`feast.constants.FEAST_CONFIG_FILE_ENV:` `str` = `'FEAST_CONFIG'`

Environmental variable to specify Feast configuration file location



## PYTHON MODULE INDEX

### f

- `feast.client`, [1](#)
- `feast.constants`, [17](#)
- `feast.data_source`, [7](#)
- `feast.entity`, [11](#)
- `feast.feature`, [15](#)
- `feast.feature_table`, [13](#)



## A

add\_feature() (feast.feature\_table.FeatureTable method), 13

apply() (feast.client.Client method), 1

apply\_entity() (feast.client.Client method), 1

apply\_feature\_table() (feast.client.Client method), 1

archive\_project() (feast.client.Client method), 1

AUTH\_PROVIDER (feast.constants.ConfigOptions attribute), 17

AUTH\_TOKEN (feast.constants.ConfigOptions attribute), 17

AZURE\_BLOB\_ACCOUNT\_ACCESS\_KEY (feast.constants.ConfigOptions attribute), 17

AZURE\_BLOB\_ACCOUNT\_NAME (feast.constants.ConfigOptions attribute), 17

## B

BATCH\_FEATURE\_REQUEST\_WAIT\_TIME\_SECONDS (feast.constants.ConfigOptions attribute), 17

BATCH\_INGESTION\_PRODUCTION\_TIMEOUT (feast.constants.ConfigOptions attribute), 17

batch\_source() (feast.feature\_table.FeatureTable property), 13

bigquery\_options() (feast.data\_source.BigQuerySource property), 7

BigQueryOptions (class in feast.data\_source), 7

BigQuerySource (class in feast.data\_source), 7

bootstrap\_servers() (feast.data\_source.KafkaOptions property), 8

## C

Client (class in feast.client), 1

CONFIG\_FEAST\_ENV\_VAR\_PREFIX (in module feast.constants), 17

CONFIG\_FILE\_DEFAULT\_DIRECTORY (in module feast.constants), 17

CONFIG\_FILE\_NAME (in module feast.constants), 17

CONFIG\_FILE\_SECTION (in module feast.constants), 17

ConfigOptions (class in feast.constants), 17

CORE\_ENABLE\_SSL (feast.constants.ConfigOptions attribute), 17

core\_secure() (feast.client.Client property), 1

CORE\_SERVER\_SSL\_CERT (feast.constants.ConfigOptions attribute), 17

CORE\_URL (feast.constants.ConfigOptions attribute), 17

core\_url() (feast.client.Client property), 1

create\_project() (feast.client.Client method), 2

created\_timestamp() (feast.entity.Entity property), 11

created\_timestamp() (feast.feature\_table.FeatureTable property), 13

created\_timestamp\_column() (feast.data\_source.DataSource property), 7

## D

DATAPROC\_CLUSTER\_NAME (feast.constants.ConfigOptions attribute), 17

DATAPROC\_EXECUTOR\_CORES (feast.constants.ConfigOptions attribute), 17

DATAPROC\_EXECUTOR\_INSTANCES (feast.constants.ConfigOptions attribute), 17

DATAPROC\_EXECUTOR\_MEMORY (feast.constants.ConfigOptions attribute), 18

DATAPROC\_PROJECT (feast.constants.ConfigOptions attribute), 18

DATAPROC\_REGION (feast.constants.ConfigOptions attribute), 18

DataSource (class in feast.data\_source), 7

date\_partition\_column() (feast.data\_source.DataSource property), 7

DATETIME\_COLUMN (in module feast.constants), 20

DEADLETTER\_PATH (*feast.constants.ConfigOptions attribute*), 18  
 delete\_feature\_table() (*feast.client.Client method*), 2  
 description() (*feast.entity.Entity property*), 11  
 dtype() (*feast.feature.Feature property*), 15

## E

EMR\_CLUSTER\_ID (*feast.constants.ConfigOptions attribute*), 18  
 EMR\_CLUSTER\_TEMPLATE\_PATH (*feast.constants.ConfigOptions attribute*), 18  
 EMR\_LOG\_LOCATION (*feast.constants.ConfigOptions attribute*), 18  
 EMR\_REGION (*feast.constants.ConfigOptions attribute*), 18  
 ENABLE\_AUTH (*feast.constants.ConfigOptions attribute*), 18  
 entities() (*feast.feature\_table.FeatureTable property*), 13  
 Entity (*class in feast.entity*), 11  
 event\_timestamp\_column() (*feast.data\_source.DataSource property*), 7

## F

feast.client  
   module, 1  
 feast.constants  
   module, 17  
 feast.data\_source  
   module, 7  
 feast.entity  
   module, 11  
 feast.feature  
   module, 15  
 feast.feature\_table  
   module, 13  
 FEAST\_CONFIG\_FILE\_ENV (*in feast.constants*), 20  
 Feature (*class in feast.feature*), 15  
 FeatureRef (*class in feast.feature*), 15  
 features() (*feast.feature\_table.FeatureTable property*), 13  
 FeatureTable (*class in feast.feature\_table*), 13  
 field\_mapping() (*feast.data\_source.DataSource property*), 8  
 file\_format() (*feast.data\_source.FileOptions property*), 8  
 file\_options() (*feast.data\_source.FileSource property*), 8  
 file\_url() (*feast.data\_source.FileOptions property*), 8

FileOptions (*class in feast.data\_source*), 8  
 FileSource (*class in feast.data\_source*), 8  
 from\_dict() (*feast.entity.Entity class method*), 11  
 from\_dict() (*feast.feature\_table.FeatureTable class method*), 13  
 from\_proto() (*feast.data\_source.BigQueryOptions class method*), 7  
 from\_proto() (*feast.data\_source.DataSource static method*), 8  
 from\_proto() (*feast.data\_source.FileOptions class method*), 8  
 from\_proto() (*feast.data\_source.KafkaOptions class method*), 8  
 from\_proto() (*feast.data\_source.KinesisOptions class method*), 9  
 from\_proto() (*feast.entity.Entity class method*), 11  
 from\_proto() (*feast.feature.Feature class method*), 15  
 from\_proto() (*feast.feature.FeatureRef class method*), 15  
 from\_proto() (*feast.feature\_table.FeatureTable class method*), 13  
 from\_str() (*feast.feature.FeatureRef class method*), 15  
 from\_yaml() (*feast.entity.Entity class method*), 11  
 from\_yaml() (*feast.feature\_table.FeatureTable class method*), 13

## G

get\_entity() (*feast.client.Client method*), 2  
 get\_feature\_table() (*feast.client.Client method*), 2  
 get\_online\_features() (*feast.client.Client method*), 2  
 get\_table\_query\_string() (*feast.data\_source.BigQuerySource method*), 7  
 GRPC\_CONNECTION\_TIMEOUT (*feast.constants.ConfigOptions attribute*), 18  
 GRPC\_CONNECTION\_TIMEOUT\_APPLY (*feast.constants.ConfigOptions attribute*), 18

## H

HISTORICAL\_FEATURE\_OUTPUT\_FORMAT (*feast.constants.ConfigOptions attribute*), 18  
 HISTORICAL\_FEATURE\_OUTPUT\_LOCATION (*feast.constants.ConfigOptions attribute*), 18

## I

ingest() (*feast.client.Client method*), 3

INGESTION\_DROP\_INVALID\_ROWS  
(*feast.constants.ConfigOptions* attribute),  
18

is\_valid() (*feast.entity.Entity* method), 11

is\_valid() (*feast.feature\_table.FeatureTable*  
method), 13

## J

JOB\_SERVICE\_ENABLE\_CONTROL\_LOOP  
(*feast.constants.ConfigOptions* attribute),  
18

JOB\_SERVICE\_ENABLE\_SSL  
(*feast.constants.ConfigOptions* attribute),  
18

job\_service\_secure() (*feast.client.Client* prop-  
erty), 3

JOB\_SERVICE\_SERVER\_SSL\_CERT  
(*feast.constants.ConfigOptions* attribute),  
18

JOB\_SERVICE\_URL (*feast.constants.ConfigOptions* at-  
tribute), 18

job\_service\_url() (*feast.client.Client* property), 4

## K

kafka\_options() (*feast.data\_source.KafkaSource*  
property), 9

KafkaOptions (class in *feast.data\_source*), 8

KafkaSource (class in *feast.data\_source*), 9

kinesis\_options()  
(*feast.data\_source.KinesisSource* property), 9

KinesisOptions (class in *feast.data\_source*), 9

KinesisSource (class in *feast.data\_source*), 9

## L

labels() (*feast.entity.Entity* property), 11

labels() (*feast.feature.Feature* property), 15

labels() (*feast.feature\_table.FeatureTable* property),  
14

last\_updated\_timestamp() (*feast.entity.Entity*  
property), 11

last\_updated\_timestamp()  
(*feast.feature\_table.FeatureTable* property), 14

list\_entities() (*feast.client.Client* method), 4

list\_feature\_tables() (*feast.client.Client*  
method), 4

list\_features\_by\_ref() (*feast.client.Client*  
method), 4

list\_projects() (*feast.client.Client* method), 4

## M

max\_age() (*feast.feature\_table.FeatureTable* prop-  
erty), 14

message\_format() (*feast.data\_source.KafkaOptions*  
property), 8

module

*feast.client*, 1

*feast.constants*, 17

*feast.data\_source*, 7

*feast.entity*, 11

*feast.feature*, 15

*feast.feature\_table*, 13

## N

name() (*feast.entity.Entity* property), 11

name() (*feast.feature.Feature* property), 15

name() (*feast.feature\_table.FeatureTable* property), 14

## O

OAuth\_AUDIENCE (*feast.constants.ConfigOptions* at-  
tribute), 18

OAuth\_CLIENT\_ID (*feast.constants.ConfigOptions* at-  
tribute), 18

OAuth\_CLIENT\_SECRET  
(*feast.constants.ConfigOptions* attribute),  
19

OAuth\_GRANT\_TYPE (*feast.constants.ConfigOptions*  
attribute), 19

OAuth\_TOKEN\_REQUEST\_URL  
(*feast.constants.ConfigOptions* attribute),  
19

## P

path() (*feast.data\_source.FileSource* property), 8

PROJECT (*feast.constants.ConfigOptions* attribute), 19

project() (*feast.client.Client* property), 4

## Q

query() (*feast.data\_source.BigQueryOptions* prop-  
erty), 7

## R

record\_format() (*feast.data\_source.KinesisOptions*  
property), 9

REDIS\_HOST (*feast.constants.ConfigOptions* attribute),  
19

REDIS\_PORT (*feast.constants.ConfigOptions* attribute),  
19

REDIS\_SSL (*feast.constants.ConfigOptions* attribute),  
19

region() (*feast.data\_source.KinesisOptions* property),  
9

REGISTRY\_PATH (*feast.constants.ConfigOptions* at-  
tribute), 19

## S

S3\_ENDPOINT\_URL (*feast.constants.ConfigOptions* at-  
tribute), 19

SERVING\_ENABLE\_SSL  
(*feast.constants.ConfigOptions* attribute), 19

serving\_secure() (*feast.client.Client* property), 4

SERVING\_SERVER\_SSL\_CERT  
(*feast.constants.ConfigOptions* attribute), 19

SERVING\_URL (*feast.constants.ConfigOptions* attribute), 19

serving\_url() (*feast.client.Client* property), 5

set\_project() (*feast.client.Client* method), 5

SourceType (class in *feast.data\_source*), 9

SPARK\_BQ\_MATERIALIZATION\_DATASET  
(*feast.constants.ConfigOptions* attribute), 19

SPARK\_BQ\_MATERIALIZATION\_PROJECT  
(*feast.constants.ConfigOptions* attribute), 19

SPARK\_HOME (*feast.constants.ConfigOptions* attribute), 19

SPARK\_INGESTION\_JAR  
(*feast.constants.ConfigOptions* attribute), 19

SPARK\_LAUNCHER (*feast.constants.ConfigOptions* attribute), 19

SPARK\_STAGING\_LOCATION  
(*feast.constants.ConfigOptions* attribute), 19

SPARK\_STANDALONE\_MASTER  
(*feast.constants.ConfigOptions* attribute), 20

STATSD\_ENABLED (*feast.constants.ConfigOptions* attribute), 20

STATSD\_HOST (*feast.constants.ConfigOptions* attribute), 20

STATSD\_PORT (*feast.constants.ConfigOptions* attribute), 20

STENCIL\_URL (*feast.constants.ConfigOptions* attribute), 20

stream\_name() (*feast.data\_source.KinesisOptions* property), 9

stream\_source() (*feast.feature\_table.FeatureTable* property), 14

## T

table\_ref() (*feast.data\_source.BigQueryOptions* property), 7

TELEMETRY (*feast.constants.ConfigOptions* attribute), 20

to\_dict() (*feast.entity.Entity* method), 11

to\_dict() (*feast.feature\_table.FeatureTable* method), 14

to\_proto() (*feast.data\_source.BigQueryOptions* method), 7

to\_proto() (*feast.data\_source.BigQuerySource* method), 7

to\_proto() (*feast.data\_source.DataSource* method), 8

to\_proto() (*feast.data\_source.FileOptions* method), 8

to\_proto() (*feast.data\_source.FileSource* method), 8

to\_proto() (*feast.data\_source.KafkaOptions* method), 9

to\_proto() (*feast.data\_source.KafkaSource* method), 9

to\_proto() (*feast.data\_source.KinesisOptions* method), 9

to\_proto() (*feast.data\_source.KinesisSource* method), 9

to\_proto() (*feast.entity.Entity* method), 11

to\_proto() (*feast.feature.Feature* method), 15

to\_proto() (*feast.feature.FeatureRef* method), 15

to\_proto() (*feast.feature\_table.FeatureTable* method), 14

to\_spec\_proto() (*feast.entity.Entity* method), 12

to\_spec\_proto() (*feast.feature\_table.FeatureTable* method), 14

to\_yaml() (*feast.entity.Entity* method), 12

to\_yaml() (*feast.feature\_table.FeatureTable* method), 14

topic() (*feast.data\_source.KafkaOptions* property), 9

## V

value\_type() (*feast.entity.Entity* property), 12

version() (*feast.client.Client* method), 5