firebolt-python-sdk

Firebolt

Jan 29, 2022

CONTENTS

Welcome to firebolt-sdk's documentation!

CHAPTER

ONE

INSTALLATION

- Requires Python >=3.7
- `pip install firebolt-sdk`

CHAPTER

TWO

CONNECTION PARAMETERS

These parameters are used to connect to a Firebolt database:

- **engine_url** - url for a Firebolt engine to make requests to. This can be retrieved from our web interface, or from the engine attribute endpoint
- **database** - the name of the database to receive queries
- **username** - Firebolt account username
- **password** - Firebolt account password

Optional parameters

- **api_endpoint** - api hostname for logging in. Defaults to `api.app.firebolt.io`.

CHAPTER

THREE

EXAMPLES

See PEP-249 for the DB API reference and specifications. An example jupyter notebook is included to illustrate the use of the Firebolt API.

CHAPTER

FOUR

OPTIONAL FEATURES

By default, firebolt-sdk uses `datetime` module to parse date and datetime values, which might be slow for a large amount of operations. In order to speed up datetime operations, it's possible to use ciso8601 package.

To install firebolt-sdk with `ciso8601` support, run `pip install firebolt-sdk[ciso8601]`

CHAPTER

FIVE

CONTRIBUTING

See: CONTRIBUTING.MD

CHAPTER

SIX

LICENSE

The Firebolt DB API is licensed under the Apache License Version 2.0 software license.

Note: This project is under active development

6.1 firebolt.async_db package

6.1.1 firebolt.async_db.connection module

class firebolt.async_db.connection.**Connection**(*engine_url: str*, *database: str*, *username: str*, *password: str*, *api_endpoint: str = 'api.app.firebolt.io'*)

Bases: firebolt.async_db.connection.BaseConnection

Firebolt asyncronous database connection class. Implements PEP 249.

Parameters

- **engine_url** – Firebolt database engine REST API url
- **database** – Firebolt database name
- **username** – Firebolt account username
- **password** – Firebolt account password
- **api_endpoint** – Optional. Firebolt API endpoint. Used for authentication.

Note: Firebolt currenly doesn't support transactions so commit and rollback methods are not implemented.

async aclose() → None
Close connection and all underlying cursors.

api_endpoint

client_class: type

cursor() → *firebolt.async_db.cursor.Cursor*

cursor_class
alias of *firebolt.async_db.cursor.Cursor*

database

engine_url

class firebolt.async_db.connection.**OverriddenHttpBackend**
Bases: httpcore.backends.auto.AutoBackend

This class is a short-term solution for TCP keep-alive issue: https://docs.aws.amazon.com/elasticloadbalancing/latest/network/network-load-balancers.html#connection-idle-timeout Since httpx creates a connection right before executing a request backend has to be overridden in order to set the socket KEEPALIVE and KEEPIDLE settings.

async connect_tcp(*host: str*, *port: int*, *timeout: Optional[float] = None*, *local_address: Optional[str] = None*) → httpcore.backends.base.AsyncNetworkStream

firebolt.async_db.connection.**async_connect_factory**(*connection_class: Type*) → Callable

async firebolt.async_db.connection.**connect**(*database: Optional[str] = None*, *username: Optional[str] = None*, *password: Optional[str] = None*, *engine_name: Optional[str] = None*, *engine_url: Optional[str] = None*, *account_name: Optional[str] = None*, *api_endpoint: str = 'api.app.firebolt.io'*) → *firebolt.async_db.connection.Connection*

Connect to Firebolt database.

Parameters

- **database** – name of the database to connect
- **username** – user name to use for authentication
- **password** – password to use for authentication
- **engine_name** – Optional The name of the engine to connect to
- **engine_url** – Optional. The engine endpoint to use
- **account_name** – For customers with multiple accounts; if None uses default.
- **api_endpoint** (*optional*) – Firebolt API endpoint. Used for authentication.

Note: Either *engine_name* or *engine_url* should be provided, but not both.

6.1.2 firebolt.async_db.cursor module

class firebolt.async_db.cursor.**Cursor**(**args: Any*, ***kwargs: Any*)
Bases: firebolt.async_db.cursor.BaseCursor

Class, responsible for executing asyncio queries to Firebolt Database. Should not be created directly, use *connection.cursor*

Parameters

- **description** – information about a single result row
- **rowcount** – the number of rows produced by last query
- **closed** – True if connection is closed, False otherwise
- **arraysize** – Read/Write, specifies the number of rows to fetch at a time with the *fetchmany()* method

connection

async execute(*query: str*, *parameters: Optional[Sequence[Union[int, float, str, datetime.datetime, datetime.date, bool, Sequence]]] = None*, *set_parameters: Optional[Dict] = None*) → int
Prepare and execute a database query. Return row count.

async executemany(*query: str*, *parameters_seq: Sequence[Sequence[Union[int, float, str, datetime.datetime, datetime.date, bool, Sequence]]]*) → int
Prepare and execute a database query against all parameter sequences provided. Return last query row count.

async fetchall() → List[List[Optional[Union[int, float, str, datetime.datetime, datetime.date, bool, list]]]]
Fetch all remaining rows of a query result.

async fetchmany(*size: Optional[int] = None*) → List[List[Optional[Union[int, float, str, datetime.datetime, datetime.date, bool, list]]]]
Fetch the next set of rows of a query result, cursor.arraysize is default size.

async fetchone() → Optional[List[Optional[Union[int, float, str, datetime.datetime, datetime.date, bool, list]]]]
Fetch the next row of a query result set.

async nextset() → Optional[bool]
Skip to the next available set, discarding any remaining rows from the current set. Returns True if operation was successful, None if there are no more sets to retrive

class firebolt.async_db.cursor.**CursorState**(*value*)
Bases: enum.Enum

An enumeration.

CLOSED = 4

DONE = 3

ERROR = 2

NONE = 1

firebolt.async_db.cursor.**check_not_closed**(*func: Callable*) → Callable
(Decorator) ensure cursor is not closed before calling method.

firebolt.async_db.cursor.**check_query_executed**(*func: Callable*) → Callable

6.1.3 Module contents

firebolt.async_db.**TimestampFromTicks**()
timestamp[, tz] -> tz's local time from POSIX timestamp.

6.2 firebolt.client package

6.2.1 firebolt.client.auth module

class firebolt.client.auth.**Auth**(*username: str*, *password: str*, *api_endpoint: str = 'api.app.firebolt.io'*)
Bases: httpx.Auth

Authentication class for Firebolt database. Gets authentication token using provided credentials and updates it when it expires

api_url

auth_flow(*request: httpx.Request*) → Generator[httpx.Request, httpx.Response, None]
Add authorization token to request headers. Overrides `httpx.Auth.auth_flow`

copy() → *firebolt.client.auth.Auth*

property expired: Optional[int]

get_new_token_generator() → Generator[httpx.Request, httpx.Response, None]
Get new token using username and password

password

requires_response_body = True

property token: Optional[str]

username

6.2.2 firebolt.client.client module

class firebolt.client.client.**AsyncClient**(**args: Any*, *account_name: Optional[str] = None*, *api_endpoint: str = 'api.app.firebolt.io'*, *auth: Optional[Union[Tuple[Union[bytes, str], Union[bytes, str]], Callable[[Request], Request], Auth]] = None*, ***kwargs: Any*)

Bases: `firebolt.client.client.FireboltClientMixin`, `httpx.AsyncClient`

An http client, based on httpx.AsyncClient, that asyncronously handles authentication for Firebolt database.

Authentication can be passed through auth keyword as a tuple or as a FireboltAuth instance

httpx.AsyncClient:

- (HttpxAsyncClient.__doc__ or “”)

account_id

class firebolt.client.client.**Client**(**args: Any*, *account_name: Optional[str] = None*, *api_endpoint: str = 'api.app.firebolt.io'*, *auth: Optional[Union[Tuple[Union[bytes, str], Union[bytes, str]], Callable[[Request], Request], Auth]] = None*, ***kwargs: Any*)

Bases: `firebolt.client.client.FireboltClientMixin`, `httpx.Client`

An http client, based on httpx.Client, that handles the authentication for Firebolt database.

Authentication can be passed through auth keyword as a tuple or as a FireboltAuth instance

httpx.Client:

- (HttpxClient.__doc__ or “”)

property account_id: str

6.2.3 firebolt.client.resource_manager_hooks module

firebolt.client.resource_manager_hooks.**log_request**(*request: httpx.Request*) → None
Hook to log http requests

firebolt.client.resource_manager_hooks.**log_response**(*response: httpx.Response*) → None
Hook to log responses to http requests

firebolt.client.resource_manager_hooks.**raise_on_4xx_5xx**(*response: httpx.Response*) → None
Hook to raise an error on http responses with codes indicating an error.

If an error is message is found raise as an ApiError

6.3 firebolt.common package

6.3.1 firebolt.common.exception module

exception firebolt.common.exception.**AccountNotFoundError**(*method_name: str*)
Bases: *firebolt.common.exception.FireboltError*

exception firebolt.common.exception.**AlreadyBoundError**
Bases: *firebolt.common.exception.FireboltEngineError*

exception firebolt.common.exception.**AttachedEngineInUseError**(*method_name: str*)
Bases: *firebolt.common.exception.FireboltDatabaseError*

exception firebolt.common.exception.**AuthenticationError**(*cause: str*, *api_endpoint: str*)
Bases: *firebolt.common.exception.FireboltError*

Firebolt authentication error. Stores error cause and authentication endpoint.

exception firebolt.common.exception.**ConnectionClosedError**
Bases: *firebolt.common.exception.ConnectionError*

exception firebolt.common.exception.**ConnectionError**
Bases: *firebolt.common.exception.FireboltError*

exception firebolt.common.exception.**CursorClosedError**(*method_name: str*)
Bases: *firebolt.common.exception.CursorError*

exception firebolt.common.exception.**CursorError**
Bases: *firebolt.common.exception.FireboltError*

exception firebolt.common.exception.**DataError**
Bases: *firebolt.common.exception.DatabaseError*

Exception raised for errors that are due to problems with the processed data like division by zero, numeric value out of range, etc.

exception firebolt.common.exception.**DatabaseError**
Bases: *firebolt.common.exception.FireboltError*

Exception raised for errors that are related to the database.

exception firebolt.common.exception.**EngineNotRunningError**
Bases: *firebolt.common.exception.FireboltEngineError*

firebolt.common.exception.**Error**
alias of *firebolt.common.exception.FireboltError*

exception firebolt.common.exception.**FireboltDatabaseError**
Bases: *firebolt.common.exception.FireboltError*

exception firebolt.common.exception.**FireboltEngineError**
Bases: *firebolt.common.exception.FireboltError*

Base error for engine errors.

exception firebolt.common.exception.**FireboltError**
Bases: Exception

exception firebolt.common.exception.**IntegrityError**
Bases: *firebolt.common.exception.DatabaseError*

Exception raised when the relational integrity of the database is affected, e.g. a foreign key check fails.

exception firebolt.common.exception.**InterfaceError**
Bases: *firebolt.common.exception.FireboltError*

Exception raised for errors that are related to the database interface rather than the database itself.

exception firebolt.common.exception.**InternalError**
Bases: *firebolt.common.exception.DatabaseError*

Exception raised when the database encounters an internal error, e.g. the cursor is not valid anymore, the transaction is out of sync, etc.

exception firebolt.common.exception.**NoAttachedDatabaseError**(*method_name: str*)
Bases: *firebolt.common.exception.FireboltEngineError*

exception firebolt.common.exception.**NotSupportedError**
Bases: *firebolt.common.exception.DatabaseError*

Exception raised when the database encounters an internal error, e.g. the cursor is not valid anymore, the transaction is out of sync, etc.

exception firebolt.common.exception.**OperationalError**
Bases: *firebolt.common.exception.DatabaseError*

Exception raised for errors that are related to the database's operation and not necessarily under the control of the programmer, e.g. an unexpected disconnect occurs, the data source name is not found, a transaction could not be processed, a memory allocation error occurred during processing, etc.

exception firebolt.common.exception.**ProgrammingError**
Bases: *firebolt.common.exception.DatabaseError*

Exception raised when the database encounters an internal error, e.g. the cursor is not valid anymore, the transaction is out of sync, etc.

exception firebolt.common.exception.**QueryError**
Bases: *firebolt.common.exception.CursorError*

exception firebolt.common.exception.**QueryNotRunError**(*method_name: str*)
Bases: *firebolt.common.exception.CursorError*

exception firebolt.common.exception.**Warning**
Bases: Exception

Exception raised for important warnings like data truncations while inserting, etc.

6.3.2 firebolt.common.settings module

class firebolt.common.settings.**Settings**(*_env_file: Optional[Union[str, os.PathLike]] = '<object object>'*, *_env_file_encoding: Optional[str] = None*, *_env_nested_delimiter: Optional[str] = None*, *_secrets_dir: Optional[Union[str, os.PathLike]] = None*, ***, *account_name: str = None*, *server: str*, *user: str*, *password: pydantic.types.SecretStr*, *default_region: str*)

Bases: pydantic.env_settings.BaseSettings

class **Config**

Bases: object

env_file = '.env'

account_name: str

default_region: str

password: pydantic.types.SecretStr

server: str

user: str

class firebolt.common.util.**AsyncJobThread**

Bases: object

Thread runner that allows running async tasks syncronously in a separate thread. Caches loop to be reused in all threads It allows running async functions syncronously inside a running event loop. Since nesting loops is not allowed, we create a separate thread for a new event loop

execute(*coro: Coroutine*) → Any

run(*coro: Coroutine*) → None

6.4 firebolt.db package

6.4.1 firebolt.db.connection module

class firebolt.db.connection.**Connection**(**args: Any*, ***kwargs: Any*)

Bases: firebolt.async_db.connection.BaseConnection

Firebolt database connection class. Implements PEP-249.

Parameters

- **engine_url** – Firebolt database engine REST API url
- **database** – Firebolt database name
- **username** – Firebolt account username
- **password** – Firebolt account password
- **api_endpoint** – Optional. Firebolt API endpoint. Used for authentication.

Note: Firebolt currenly doesn't support transactions so commit and rollback methods are not implemented.

---

api_endpoint

client_class: type

close() → None
Close connection and all underlying cursors.

cursor() → *firebolt.db.cursor.Cursor*

cursor_class
alias of *firebolt.db.cursor.Cursor*

database

engine_url

6.4.2 firebolt.db.cursor module

class firebolt.db.cursor.**Cursor**(**args: Any*, ***kwargs: Any*)
Bases: firebolt.async_db.cursor.BaseCursor

Class, responsible for executing queries to Firebolt Database. Should not be created directly, use *connection.cursor*

Parameters

- **description** – Information about a single result row
- **rowcount** – The number of rows produced by last query
- **closed** – True if connection is closed, False otherwise
- **arraysize** – Read/Write,
- **time** (*specifies the number of rows to fetch at a*) –

:param with *fetchmany()* method:

connection

execute(*query: str*, *parameters: Optional[Sequence[Union[int, float, str, datetime.datetime, datetime.date, bool, Sequence]]] = None*, *set_parameters: Optional[Dict] = None*) → int
Prepare and execute a database query. Return row count.

executemany(*query: str*, *parameters_seq: Sequence[Sequence[Union[int, float, str, datetime.datetime, datetime.date, bool, Sequence]]]*) → int
Prepare and execute a database query against all parameter sequences provided. Return last query row count.

fetchall() → List[List[Optional[Union[int, float, str, datetime.datetime, datetime.date, bool, list]]]]
Fetch all remaining rows of a query result.

fetchmany(*size: Optional[int] = None*) → List[List[Optional[Union[int, float, str, datetime.datetime, datetime.date, bool, list]]]]
Fetch the next set of rows of a query result, cursor.arraysize is default size.

fetchone() → Optional[List[Optional[Union[int, float, str, datetime.datetime, datetime.date, bool, list]]]]
Fetch the next row of a query result set.

nextset() → Optional[bool]
Skip to the next available set, discarding any remaining rows from the current set. Returns True if operation was successful, None if there are no more sets to retrive

6.5 firebolt.model package

6.5.1 firebolt.model.database module

class firebolt.model.database.**Database**(*, *name: firebolt.model.database.ConstrainedStrValue*, *compute_region_id: firebolt.model.region.RegionKey*, *id: firebolt.model.database.DatabaseKey = None*, *description: firebolt.model.database.ConstrainedStrValue = None*, *emoji: firebolt.model.database.ConstrainedStrValue = None*, *current_status: str = None*, *health_status: str = None*, *data_size_full: int = None*, *data_size_compressed: int = None*, *is_system_database: bool = None*, *storage_bucket_name: str = None*, *create_time: datetime.datetime = None*, *create_actor: str = None*, *last_update_time: datetime.datetime = None*, *last_update_actor: str = None*, *desired_status: str = None*)

Bases: firebolt.model.FireboltBaseModel

A Firebolt database.

Databases belong to a region and have a description, but otherwise are not configurable.

attach_to_engine(*engine:* Engine, *is_default_engine: bool = False*) → Binding

Attach an engine to this database.

Parameters

- **engine** – The engine to attach.
- **is_default_engine** – Whether this engine should be used as default for this database. Only one engine can be set as default for a single database. This will overwrite any existing default.

compute_region_key: RegionKey

create_actor: Optional[str]

create_time: Optional[datetime]

current_status: Optional[str]

data_size_compressed: Optional[int]

data_size_full: Optional[int]

property database_id: Optional[str]

database_key: Optional[DatabaseKey]

delete() → *firebolt.model.database.Database*

Delete a database from Firebolt.

Raises an error if there are any attached engines.

description: Optional[str]

desired_status: Optional[str]

emoji: Optional[str]

get_attached_engines() → List[*Engine*]

Get a list of engines that are attached to this database.

health_status: Optional[str]

is_system_database: Optional[bool]

last_update_actor: Optional[str]

last_update_time: Optional[datetime]

name: str

classmethod parse_obj_with_service(*obj: Any*, *database_service:* DatabaseService) → *Database*

storage_bucket_name: Optional[str]

6.5.2 firebolt.model.engine module

class firebolt.model.engine.**Engine**(*, *name: firebolt.model.engine.ConstrainedStrValue*, *compute_region_id: firebolt.model.region.RegionKey*, *settings:* firebolt.model.engine.EngineSettings, *id: firebolt.model.engine.EngineKey = None*, *description: str = None*, *emoji: str = None*, *current_status:* firebolt.service.types.EngineStatus *= None*, *current_status_summary:* firebolt.service.types.EngineStatusSummary *= None*, *latest_revision_id: firebolt.model.engine_revision.EngineRevisionKey = None*, *endpoint: str = None*, *endpoint_serving_revision_id: firebolt.model.engine_revision.EngineRevisionKey = None*, *create_time: datetime.datetime = None*, *create_actor: str = None*, *last_update_time: datetime.datetime = None*, *last_update_actor: str = None*, *last_use_time: datetime.datetime = None*, *desired_status: str = None*, *health_status: str = None*, *endpoint_desired_revision_id: firebolt.model.engine_revision.EngineRevisionKey = None*)

Bases: firebolt.model.FireboltBaseModel

A Firebolt engine. Responsible for performing work (queries, data ingestion).

Engines are configured in *Settings* and in *EngineRevisionSpecification*.

attach_to_database(*database:* firebolt.model.database.Database, *is_default_engine: bool = False*) → firebolt.model.binding.Binding

Attach this engine to a database.

Parameters

- **database** – Database to which the engine will be attached.
- **is_default_engine** – Whether this engine should be used as default for this database. Only one engine can be set as default for a single database. This will overwrite any existing default.

compute_region_key: RegionKey

create_actor: Optional[str]

create_time: Optional[datetime]

current_status: Optional[*EngineStatus*]

current_status_summary: Optional[*EngineStatusSummary*]

property database: Optional[*firebolt.model.database.Database*]

delete() → *firebolt.model.engine.Engine*

Delete an Engine from Firebolt.

description: Optional[str]

desired_status: Optional[str]

emoji: Optional[str]

endpoint: Optional[str]

endpoint_desired_revision_key: Optional[EngineRevisionKey]

endpoint_serving_revision_key: Optional[EngineRevisionKey]

property engine_id: str

get_connection() → *firebolt.db.connection.Connection*
Get a connection to the attached database, for running queries.

get_latest() → *firebolt.model.engine.Engine*
Get an up-to-date instance of the Engine from Firebolt.

health_status: Optional[str]

key: Optional[EngineKey]

last_update_actor: Optional[str]

last_update_time: Optional[datetime]

last_use_time: Optional[datetime]

latest_revision_key: Optional[EngineRevisionKey]

name: str

classmethod parse_obj_with_service(*obj: Any*, *engine_service:* EngineService) → *Engine*

settings: *EngineSettings*

start(*wait_for_startup: bool = True*, *wait_timeout_seconds: int = 3600*, *verbose: bool = False*) → *firebolt.model.engine.Engine*
Start an engine. If it's already started, do nothing.

Parameters

- **wait_for_startup** – If True, wait for startup to complete. If false, return immediately after requesting startup.
- **wait_timeout_seconds** – Number of seconds to wait for startup to complete before raising a TimeoutError.
- **verbose** – If True, print dots periodically while waiting for engine startup. If false, do not print any dots.

Returns The updated Engine from Firebolt.

stop(*wait_for_stop: bool = False*, *wait_timeout_seconds: int = 3600*) → *firebolt.model.engine.Engine*
Stop an Engine running on Firebolt.

class firebolt.model.engine.**EngineSettings**(*, *preset: str*, *auto_stop_delay_duration: firebolt.model.engine.ConstrainedStrValue*, *minimum_logging_level: str*, *is_read_only: bool*, *warm_up: str*)

Bases: firebolt.model.FireboltBaseModel

Engine Settings.

See Also: *EngineRevisionSpecification* which also contains engine configuration.

auto_stop_delay_duration: str

classmethod default(*engine_type:* firebolt.service.types.EngineType = *EngineType.GENERAL_PURPOSE*, *auto_stop_delay_duration: str = '1200s'*, *warm_up:* firebolt.service.types.WarmupMethod = *WarmupMethod.PRELOAD_INDEXES*) → *firebolt.model.engine.EngineSettings*

is_read_only: bool

minimum_logging_level: str

preset: str

warm_up: str

firebolt.model.engine.**check_attached_to_database**(*func: Callable*) → Callable
(Decorator) Ensure the engine is attached to a database.

firebolt.model.engine.**wait**(*seconds: int*, *timeout_time: float*, *error_message: str*, *verbose: bool*) → None

6.5.3 firebolt.model.engine_revision module

class firebolt.model.engine_revision.**EngineRevision**(*, *specification:* firebolt.model.engine_revision.EngineRevisionSpecification, *id: firebolt.model.engine_revision.EngineRevisionKey = None*, *current_status: str = None*, *create_time: datetime.datetime = None*, *create_actor: str = None*, *last_update_time: datetime.datetime = None*, *last_update_actor: str = None*, *desired_status: str = None*, *health_status: str = None*)

Bases: firebolt.model.FireboltBaseModel

A Firebolt Engine revision, which contains a Specification (instance types, counts).

As engines are updated with new settings, revisions are created.

create_actor: Optional[str]

create_time: Optional[datetime.datetime]

current_status: Optional[str]

desired_status: Optional[str]

health_status: Optional[str]

key: Optional[firebolt.model.engine_revision.EngineRevisionKey]

last_update_actor: Optional[str]

last_update_time: Optional[datetime.datetime]

specification: *firebolt.model.engine_revision.EngineRevisionSpecification*

class firebolt.model.engine_revision.**EngineRevisionSpecification**(*, *db_compute_instances_type_id: firebolt.model.instance_type.InstanceTypeKey*, *db_compute_instances_count: pydantic.types.PositiveInt*, *db_compute_instances_use_spot: bool = False*, *db_version: str = ''*, *proxy_instances_type_id: firebolt.model.instance_type.InstanceTypeKey*, *proxy_instances_count: pydantic.types.PositiveInt = 1*, *proxy_version: str = ''*)

Bases: firebolt.model.FireboltBaseModel

An EngineRevision Specification.

Notably, it determines which instance types and how many of them its Engine gets.

See Also: *Settings*, which also contains engine configuration.

db_compute_instances_count: pydantic.types.PositiveInt

db_compute_instances_type_key: firebolt.model.instance_type.InstanceTypeKey

db_compute_instances_use_spot: bool

db_version: str

proxy_instances_count: pydantic.types.PositiveInt

proxy_instances_type_key: firebolt.model.instance_type.InstanceTypeKey

proxy_version: str

6.5.4 firebolt.model.instance_type module

class firebolt.model.instance_type.**InstanceType**(*, *id: firebolt.model.instance_type.InstanceTypeKey*, *name: str*, *is_spot_available: bool = None*, *cpu_virtual_cores_count: int = None*, *memory_size_bytes: str = None*, *storage_size_bytes: str = None*, *price_per_hour_cents: float = None*, *create_time: datetime.datetime = None*, *last_update_time: datetime.datetime = None*)

Bases: firebolt.model.FireboltBaseModel

cpu_virtual_cores_count: Optional[int]

create_time: Optional[datetime.datetime]

is_spot_available: Optional[bool]

key: firebolt.model.instance_type.InstanceTypeKey

last_update_time: Optional[datetime.datetime]

memory_size_bytes: Optional[str]

name: str

price_per_hour_cents: Optional[float]

storage_size_bytes: Optional[str]

6.5.5 firebolt.model.provider module

class firebolt.model.provider.**Provider**(*, *id: str*, *name: str*, *create_time: datetime.datetime = None*, *display_name: str = None*, *last_update_time: datetime.datetime = None*)

Bases: firebolt.model.FireboltBaseModel

create_time: Optional[datetime.datetime]

display_name: Optional[str]

last_update_time: Optional[datetime.datetime]

name: str

provider_id: str

6.5.6 firebolt.model.region module

class firebolt.model.region.**Region**(*, *id: firebolt.model.region.RegionKey*, *name: str*, *display_name: str = None*, *create_time: datetime.datetime = None*, *last_update_time: datetime.datetime = None*)

Bases: firebolt.model.FireboltBaseModel

create_time: Optional[datetime.datetime]

display_name: Optional[str]

key: firebolt.model.region.RegionKey

last_update_time: Optional[datetime.datetime]

name: str

6.6 firebolt.service package

6.6.1 firebolt.service.database module

class firebolt.service.database.**DatabaseService**(*resource_manager:* firebolt.service.manager.ResourceManager)

Bases: firebolt.service.base.BaseService

create(*name: str*, *region: Optional[str] = None*, *description: Optional[str] = None*) → *firebolt.model.database.Database*

Create a new Database on Firebolt.

Parameters

- **name** – Name of the database.
- **region** – Region name in which to create the database.

Returns The newly created Database.

get(*id_: str*) → *firebolt.model.database.Database*

Get a Database from Firebolt by its id.

get_by_name(*name: str*) → *firebolt.model.database.Database*
Get a Database from Firebolt by its name.

get_id_by_name(*name: str*) → str
Get a Database id from Firebolt by its name.

get_many(*name_contains: Optional[str] = None*, *attached_engine_name_eq: Optional[str] = None*, *attached_engine_name_contains: Optional[str] = None*, *order_by: Optional[Union[str, firebolt.service.types.DatabaseOrder]] = None*) → List[*firebolt.model.database.Database*]
Get a list of databases on Firebolt.

Parameters

- **name_contains** – Filter for databases with a name containing this substring.
- **attached_engine_name_eq** – Filter for databases by an exact engine name.
- **attached_engine_name_contains** – Filter for databases by engines with a name containing this substring.
- **order_by** – Method by which to order the results.

:param See *firebolt.service.types.DatabaseOrder*.:

Returns A list of databases matching the filters.

6.6.2 firebolt.service.engine module

class firebolt.service.engine.**EngineService**(*resource_manager: firebolt.service.manager.ResourceManager*)
Bases: firebolt.service.base.BaseService

create(*name: str*, *region: Optional[Union[str, firebolt.model.region.Region]] = None*, *engine_type: Union[str, firebolt.service.types.EngineType] = EngineType.GENERAL_PURPOSE*, *scale: int = 2*, *spec: str = 'i3.4xlarge'*, *auto_stop: int = 20*, *warmup: Union[str, firebolt.service.types.WarmupMethod] = WarmupMethod.PRELOAD_INDEXES*, *description: str = ''*) → *firebolt.model.engine.Engine*
Create a new Engine.

Parameters

- **name** – An identifier that specifies the name of the engine.
- **region** – The AWS region in which the engine runs.
- **engine_type** – The engine type. GENERAL_PURPOSE or DATA_ANALYTICS
- **scale** – The number of compute instances on the engine. The scale can be any int from 1 to 128.
- **spec** – The AWS EC2 instance type.
- **auto_stop** – The amount of time (in minutes)
- **stops.** (*after which the engine automatically*) –
- **warmup** – The warmup method that should be used.

  MINIMAL - On-demand loading (both indexes and tables' data).

  PRELOAD_INDEXES - Load indexes only.

  PRELOAD_ALL_DATA - Full data auto-load (both indexes and table data - full warmup).

- **description** – A short description of the engine's purpose.

Returns Engine with the specified settings.

get(*id_: str*) → *firebolt.model.engine.Engine*
Get an Engine from Firebolt by its id.

get_by_ids(*ids: List[str]*) → List[*firebolt.model.engine.Engine*]
Get multiple Engines from Firebolt by their ids.

get_by_name(*name: str*) → *firebolt.model.engine.Engine*
Get an Engine from Firebolt by its name.

get_many(*name_contains: str*, *current_status_eq: str*, *current_status_not_eq: str*, *region_eq: str*, *order_by: Union[str,* firebolt.service.types.EngineOrder*]*) → List[*firebolt.model.engine.Engine*]
Get a list of engines on Firebolt.

Parameters

- **name_contains** – Filter for engines with a name containing this substring.
- **current_status_eq** – Filter for engines with this status.
- **current_status_not_eq** – Filter for engines that do not have this status.
- **region_eq** – Filter for engines by region.
- **order_by** – Method by which to order the results. See [EngineOrder].

Returns A list of engines matching the filters.

6.6.3 firebolt.service.instance_type module

class firebolt.service.instance_type.**InstanceTypeService**(*resource_manager:* firebolt.service.manager.ResourceManager)

Bases: firebolt.service.base.BaseService

get_by_key(*instance_type_key: firebolt.model.instance_type.InstanceTypeKey*) → *firebolt.model.instance_type.InstanceType*
Get an instance type by key.

get_by_name(*instance_type_name: str*, *region_name: Optional[str] = None*) → *firebolt.model.instance_type.InstanceType*
Get an instance type by name.

Parameters

- **instance_type_name** – Name of the instance (eg. "i3.4xlarge").
- **region_name** – Name of the AWS region from which to get the instance. If not provided, use the default region name from the client.

Returns The requested instance type.

property instance_types: List[*firebolt.model.instance_type.InstanceType*]
List of instance types available on Firebolt.

property instance_types_by_key: Dict[firebolt.model.instance_type.InstanceTypeKey, *firebolt.model.instance_type.InstanceType*]
Dict of {InstanceTypeKey to InstanceType}

property instance_types_by_name:
Dict[firebolt.service.instance_type.InstanceTypeLookup,
***firebolt.model.instance_type.InstanceType*]**
Dict of {InstanceTypeLookup to InstanceType}

6.6.4 firebolt.service.manager module

class firebolt.service.manager.**ResourceManager**(*settings: Optional[*firebolt.common.settings.Settings*] = None*, *account_name: Optional[str] = None*)

Bases: `object`

ResourceManager to access various Firebolt resources:

- databases
- engines
- bindings (the bindings between an engine and a database)
- engine revisions (versions of an engine)

Also provides listings of:

- regions (AWS regions in which engines can run)
- instance types (AWS instance types which engines can use)

6.6.5 firebolt.service.provider module

firebolt.service.provider.**get_provider_id**(*client:* firebolt.client.client.Client) → str
Get the AWS provider_id.

6.6.6 firebolt.service.region module

class firebolt.service.region.**RegionService**(*resource_manager:* firebolt.service.manager.ResourceManager)

Bases: `firebolt.service.base.BaseService`

property default_region: ***firebolt.model.region.Region***
Default AWS Region, could be provided from environment.

get_by_id(*id_: str*) → *firebolt.model.region.Region*
Get an AWS Region by region_id.

get_by_key(*key: firebolt.model.region.RegionKey*) → *firebolt.model.region.Region*
Get an AWS Region by its key.

get_by_name(*name: str*) → *firebolt.model.region.Region*
Get an AWS region by its name (eg. us-east-1).

property regions: List[*firebolt.model.region.Region*]
List of available AWS Regions on Firebolt.

property regions_by_key: Dict[firebolt.model.region.RegionKey,
***firebolt.model.region.Region*]**
Dict of {RegionKey to Region}

property regions_by_name: Dict[str, *firebolt.model.region.Region*]
Dict of {RegionLookup to Region}

6.6.7 firebolt.service.types module

class firebolt.service.types.**DatabaseOrder**(*value*)
Bases: enum.Enum

An enumeration.

DATABASE_ORDER_COMPUTE_REGION_ID_ASC = 'DATABASE_ORDER_COMPUTE_REGION_ID_ASC'

DATABASE_ORDER_COMPUTE_REGION_ID_DESC = 'DATABASE_ORDER_COMPUTE_REGION_ID_DESC'

DATABASE_ORDER_CREATE_ACTOR_ASC = 'DATABASE_ORDER_CREATE_ACTOR_ASC'

DATABASE_ORDER_CREATE_ACTOR_DESC = 'DATABASE_ORDER_CREATE_ACTOR_DESC'

DATABASE_ORDER_CREATE_TIME_ASC = 'DATABASE_ORDER_CREATE_TIME_ASC'

DATABASE_ORDER_CREATE_TIME_DESC = 'DATABASE_ORDER_CREATE_TIME_DESC'

DATABASE_ORDER_DATA_SIZE_COMPRESSED_ASC = 'DATABASE_ORDER_DATA_SIZE_COMPRESSED_ASC'

DATABASE_ORDER_DATA_SIZE_COMPRESSED_DESC =
'DATABASE_ORDER_DATA_SIZE_COMPRESSED_DESC'

DATABASE_ORDER_DATA_SIZE_FULL_ASC = 'DATABASE_ORDER_DATA_SIZE_FULL_ASC'

DATABASE_ORDER_DATA_SIZE_FULL_DESC = 'DATABASE_ORDER_DATA_SIZE_FULL_DESC'

DATABASE_ORDER_LAST_UPDATE_ACTOR_ASC = 'DATABASE_ORDER_LAST_UPDATE_ACTOR_ASC'

DATABASE_ORDER_LAST_UPDATE_ACTOR_DESC = 'DATABASE_ORDER_LAST_UPDATE_ACTOR_DESC'

DATABASE_ORDER_LAST_UPDATE_TIME_ASC = 'DATABASE_ORDER_LAST_UPDATE_TIME_ASC'

DATABASE_ORDER_LAST_UPDATE_TIME_DESC = 'DATABASE_ORDER_LAST_UPDATE_TIME_DESC'

DATABASE_ORDER_NAME_ASC = 'DATABASE_ORDER_NAME_ASC'

DATABASE_ORDER_NAME_DESC = 'DATABASE_ORDER_NAME_DESC'

DATABASE_ORDER_UNSPECIFIED = 'DATABASE_ORDER_UNSPECIFIED'

class firebolt.service.types.**EngineOrder**(*value*)
Bases: enum.Enum

An enumeration.

ENGINE_ORDER_COMPUTE_REGION_ID_ASC = 'ENGINE_ORDER_COMPUTE_REGION_ID_ASC'

ENGINE_ORDER_COMPUTE_REGION_ID_DESC = 'ENGINE_ORDER_COMPUTE_REGION_ID_DESC'

ENGINE_ORDER_CREATE_ACTOR_ASC = 'ENGINE_ORDER_CREATE_ACTOR_ASC'

ENGINE_ORDER_CREATE_ACTOR_DESC = 'ENGINE_ORDER_CREATE_ACTOR_DESC'

ENGINE_ORDER_CREATE_TIME_ASC = 'ENGINE_ORDER_CREATE_TIME_ASC'

ENGINE_ORDER_CREATE_TIME_DESC = 'ENGINE_ORDER_CREATE_TIME_DESC'

ENGINE_ORDER_CURRENT_STATUS_ASC = 'ENGINE_ORDER_CURRENT_STATUS_ASC'

ENGINE_ORDER_CURRENT_STATUS_DESC = 'ENGINE_ORDER_CURRENT_STATUS_DESC'

ENGINE_ORDER_LAST_UPDATE_ACTOR_ASC = 'ENGINE_ORDER_LAST_UPDATE_ACTOR_ASC'

ENGINE_ORDER_LAST_UPDATE_ACTOR_DESC = 'ENGINE_ORDER_LAST_UPDATE_ACTOR_DESC'

ENGINE_ORDER_LAST_UPDATE_TIME_ASC = 'ENGINE_ORDER_LAST_UPDATE_TIME_ASC'

ENGINE_ORDER_LAST_UPDATE_TIME_DESC = 'ENGINE_ORDER_LAST_UPDATE_TIME_DESC'

ENGINE_ORDER_LATEST_REVISION_CURRENT_STATUS_ASC = 'ENGINE_ORDER_LATEST_REVISION_CURRENT_STATUS_ASC'

ENGINE_ORDER_LATEST_REVISION_CURRENT_STATUS_DESC = 'ENGINE_ORDER_LATEST_REVISION_CURRENT_STATUS_DESC'

ENGINE_ORDER_LATEST_REVISION_SPECIFICATION_DB_COMPUTE_INSTANCES_COUNT_ASC = 'ENGINE_ORDER_LATEST_REVISION_SPECIFICATION_DB_COMPUTE_INSTANCES_COUNT_ASC'

ENGINE_ORDER_LATEST_REVISION_SPECIFICATION_DB_COMPUTE_INSTANCES_COUNT_DESC = 'ENGINE_ORDER_LATEST_REVISION_SPECIFICATION_DB_COMPUTE_INSTANCES_COUNT_DESC'

ENGINE_ORDER_LATEST_REVISION_SPECIFICATION_DB_COMPUTE_INSTANCES_TYPE_ID_ASC = 'ENGINE_ORDER_LATEST_REVISION_SPECIFICATION_DB_COMPUTE_INSTANCES_TYPE_ID_ASC'

ENGINE_ORDER_LATEST_REVISION_SPECIFICATION_DB_COMPUTE_INSTANCES_TYPE_ID_DESC = 'ENGINE_ORDER_LATEST_REVISION_SPECIFICATION_DB_COMPUTE_INSTANCES_TYPE_ID_DESC'

ENGINE_ORDER_NAME_ASC = 'ENGINE_ORDER_NAME_ASC'

ENGINE_ORDER_NAME_DESC = 'ENGINE_ORDER_NAME_DESC'

ENGINE_ORDER_UNSPECIFIED = 'ENGINE_ORDER_UNSPECIFIED'

class firebolt.service.types.**EngineStatus**(*value*)

Bases: enum.Enum

Detailed engine status.

See: https://api.dev.firebolt.io/devDocs#operation/coreV1GetEngine

ENGINE_STATUS_CREATED = 'ENGINE_STATUS_CREATED'
Engine status was created.

ENGINE_STATUS_DELETED = 'ENGINE_STATUS_DELETED'
Engine is soft-deleted.

ENGINE_STATUS_PROVISIONING_FAILED = 'ENGINE_STATUS_PROVISIONING_FAILED'
Engine initialization failed due to error.

ENGINE_STATUS_PROVISIONING_FINISHED = 'ENGINE_STATUS_PROVISIONING_FINISHED'
Engine initialization was finished successfully.

ENGINE_STATUS_PROVISIONING_PENDING = 'ENGINE_STATUS_PROVISIONING_PENDING'
Engine initialization request was sent.

ENGINE_STATUS_PROVISIONING_STARTED = 'ENGINE_STATUS_PROVISIONING_STARTED'
Engine initialization request was received and initialization process started.

ENGINE_STATUS_RUNNING_IDLE = 'ENGINE_STATUS_RUNNING_IDLE'
Engine is initialized, but there are no running or starting engine revisions.

ENGINE_STATUS_RUNNING_REVISIONS_TERMINATING = 'ENGINE_STATUS_RUNNING_REVISIONS_TERMINATING'
Engine is initialized, all child revisions are being terminated.

ENGINE_STATUS_RUNNING_REVISION_CHANGE_FAILED = 'ENGINE_STATUS_RUNNING_REVISION_CHANGE_FAILED'
Engine is ready (serves an engine revision), replacement revision failed to provision or start.

ENGINE_STATUS_RUNNING_REVISION_CHANGING = 'ENGINE_STATUS_RUNNING_REVISION_CHANGING'
Engine is ready (serves an engine revision), zero-downtime replacement revision is starting.

ENGINE_STATUS_RUNNING_REVISION_RESTARTING = 'ENGINE_STATUS_RUNNING_REVISION_RESTARTING'
Engine is initialized, replacement of the revision with a downtime is in progress.

ENGINE_STATUS_RUNNING_REVISION_RESTART_FAILED = 'ENGINE_STATUS_RUNNING_REVISION_RESTART_FAILED'
Engine is initialized, replacement revision failed to provision or start.

ENGINE_STATUS_RUNNING_REVISION_SERVING = 'ENGINE_STATUS_RUNNING_REVISION_SERVING'
Engine is ready (serves an engine revision).

ENGINE_STATUS_RUNNING_REVISION_STARTING = 'ENGINE_STATUS_RUNNING_REVISION_STARTING'
Engine is initialized, there are no running engine revision but it's starting.

ENGINE_STATUS_RUNNING_REVISION_STARTUP_FAILED = 'ENGINE_STATUS_RUNNING_REVISION_STARTUP_FAILED'
Engine is initialized, initial revision is failed to provision or start.

ENGINE_STATUS_TERMINATION_F = 'ENGINE_STATUS_TERMINATION_FAILED'
Engine termination failed.

ENGINE_STATUS_TERMINATION_FIN = 'ENGINE_STATUS_TERMINATION_FINISHED'
Engine termination finished.

ENGINE_STATUS_TERMINATION_PENDING = 'ENGINE_STATUS_TERMINATION_PENDING'
Engine termination request was sent.

ENGINE_STATUS_TERMINATION_ST = 'ENGINE_STATUS_TERMINATION_STARTED'
Engine termination started.

ENGINE_STATUS_UNSPECIFIED = 'ENGINE_STATUS_UNSPECIFIED'
Logical record is created, however underlying infrastructure is not initialized. In other words this means that engine is stopped.

class firebolt.service.types.**EngineStatusSummary**(*value*)
Bases: enum.Enum

Engine summary status.

See: https://api.dev.firebolt.io/devDocs#operation/coreV1GetEngine

ENGINE_STATUS_SUMMARY_DELETED = 'ENGINE_STATUS_SUMMARY_DELETED'
Infrastructure is terminated, engine data is deleted.

ENGINE_STATUS_SUMMARY_DELETING = 'ENGINE_STATUS_SUMMARY_DELETING'
Termination is in progress. All infrastructure that belongs to this engine will be completely destroyed.

ENGINE_STATUS_SUMMARY_FAILED = 'ENGINE_STATUS_SUMMARY_FAILED'
Failed to start or stop. This status only indicates that there were issues during provisioning operations. If engine enters this status, all infrastructure should be stopped/terminated already.

ENGINE_STATUS_SUMMARY_REPAIRING = 'ENGINE_STATUS_SUMMARY_REPAIRING'
Underlying infrastructure has issues and is being repaired. Engine is still running, but it's not fully healthy and some queries may fail.

ENGINE_STATUS_SUMMARY_RESTARTING = 'ENGINE_STATUS_SUMMARY_RESTARTING'
Hard restart (full stop/start cycle) is in progress. Underlying infrastructure is being recreated.

ENGINE_STATUS_SUMMARY_RESTARTING_INITIALIZING = 'ENGINE_STATUS_SUMMARY_RESTARTING_INITIALIZING'
Hard restart (full stop/start cycle) is in progress. Underlying infrastructure is ready, waiting for PackDB cluster to initialize and start. This status is logically the same as ENGINE_STATUS_SUMMARY_STARTING_INITIALIZING, but used during restart cycle.

ENGINE_STATUS_SUMMARY_RUNNING = 'ENGINE_STATUS_SUMMARY_RUNNING'
Fully started. Engine is ready to serve requests.

ENGINE_STATUS_SUMMARY_STARTING = 'ENGINE_STATUS_SUMMARY_STARTING'
Provisioning process is in progress. We are creating cloud infra for this engine.

ENGINE_STATUS_SUMMARY_STARTING_INITIALIZING = 'ENGINE_STATUS_SUMMARY_STARTING_INITIALIZING'
Provisioning process is complete. We are now waiting for PackDB cluster to initialize and start.

ENGINE_STATUS_SUMMARY_STOPPED = 'ENGINE_STATUS_SUMMARY_STOPPED'
Fully stopped.

ENGINE_STATUS_SUMMARY_STOPPING = 'ENGINE_STATUS_SUMMARY_STOPPING'
Stop is in progress.

ENGINE_STATUS_SUMMARY_UNSPECIFIED = 'ENGINE_STATUS_SUMMARY_UNSPECIFIED'
Status unspecified

ENGINE_STATUS_SUMMARY_UPGRADING = 'ENGINE_STATUS_SUMMARY_UPGRADING'
Version of the PackDB is changing. This is zero downtime operation that does not affect engine work. This status is reserved for future use (not used fow now).

class firebolt.service.types.**EngineType**(*value*)
Bases: enum.Enum

An enumeration.

DATA_ANALYTICS = 'DATA_ANALYTICS'

GENERAL_PURPOSE = 'GENERAL_PURPOSE'

api_settings_preset_name

class firebolt.service.types.**WarmupMethod**(*value*)
Bases: enum.Enum

An enumeration.

MINIMAL = 'MINIMAL'

PRELOAD_ALL_DATA = 'PRELOAD_ALL_DATA'

PRELOAD_INDEXES = 'PRELOAD_INDEXES'

api_name

6.7 Indices and tables

- genindex
- modindex
- search

PYTHON MODULE INDEX

f

INDEX

D

E

F

G

H

I

K

L

M

N

O

P

Q

R

S

T

U

W