sprockets-dynamodb

Release 2.3.0

Jan 24, 2018

Contents

An asynchronous DynamoDB client and mixin for Tornado applications

CHAPTER 1

Installation

`sprockets-dynamodb` is available on the Python package index and is installable via pip:

```
pip install sprockets-dynamodb
```

CHAPTER 2

Documentation

Documentation is available at sprockets-dynamodb.readthedocs.io.

CHAPTER 3

Configuration

The following table details the environment variable configuration options.

Variable	Definition	Default
`DYNAMODB_ENDPOINT`	Override the default DynamoDB HTTP endpoint	
`DYNAMODB_MAX_CLIENTS`	Maximum number of concurrent DynamoDB clients/requests per process	`100`
`DYNAMODB_MAX_RETRIES`	Maximum number retries for transient errors	`3`
`DYANMODB_NO_CREDS_RATE_LIMIT`	If set to true, a `sprockets_dynamodb.NoCredentialsException` will return a 429 response when using `sprockets_dynamodb.DynamoDBMixin`	

3.1 Mixin Configuration

The `sprockets_dynamodb.DynamoDBMixin` class will automatically raise `HTTPError` responses for different classes of errors coming from DynamoDB. In addition it will attempt to work with the Sprockets InfluxDB client to instrument all DynamoDB requests, submitting per request measurements to InfluxDB. It will attempt to automatically tag measurements with the application/service name if the `SERVICE` environment variable is set. It will also tag the measurement if the `ENVIRONMENT` environment variable is set with the environment that the application is running in. Finally, if you are using the Sprockets Correlation Mixin, measurements will automatically be tagged with the correlation ID for a request.

CHAPTER 4

Requirements

- Tornado
- tornado-aws

CHAPTER 5

Version History

Available at https://sprockets-dynamodb.readthedocs.org/en/latest/history.html

5.1 API Documentation

class `sprockets_dynamodb.client.`**`Client`**(***kwargs*)

Asynchronous DynamoDB Client

Parameters

- **`region`** (*`str`*) – AWS region to send requests to
- **`access_key`** (*`str`*) – AWS access key. If unspecified, this defaults to the `AWS_ACCESS_KEY_ID` environment variable and will fall back to using the AWS CLI credentials file. See `tornado_aws.client.AsyncAWSClient` for more details.
- **`secret_key`** (*`str`*) – AWS secret used to secure API calls. If unspecified, this defaults to the `AWS_SECRET_ACCESS_KEY` environment variable and will fall back to using the AWS CLI credentials as described in `tornado_aws.client.AsyncAWSClient`.
- **`profile`** (*`str`*) – optional profile to use in AWS API calls. If unspecified, this defaults to the `AWS_DEFAULT_PROFILE` environment variable or `default` if unset.
- **`endpoint`** (*`str`*) – DynamoDB endpoint to contact. If unspecified, the default is determined by the region.
- **`max_clients`** (*`int`*) – optional maximum number of HTTP requests that may be performed in parallel.
- **`max_retries`** (*`int`*) – Maximum number of times to retry a request when if fails under certain conditions. Can also be set with the `DYNAMODB_MAX_RETRIES` environment variable.
- **`instrumentation_callback`** (*`method`*) – A method that is invoked with a list of measurements that were collected during the execution of an individual action.

- **on_error_callback** (*method*) – A method that is invoked when there is a request exception that can not automatically be retried or the maximum number of retries has been exceeded for a request.

Any of the methods invoked in the client can raise the following exceptions:

- *sprockets_dynamodb.exceptions.DynamoDBException*
- *sprockets_dynamodb.exceptions.ConfigNotFound*
- *sprockets_dynamodb.exceptions.NoCredentialsError*
- *sprockets_dynamodb.exceptions.NoProfileError*
- *sprockets_dynamodb.exceptions.TimeoutException*
- *sprockets_dynamodb.exceptions.RequestException*
- *sprockets_dynamodb.exceptions.InternalFailure*
- *sprockets_dynamodb.exceptions.LimitExceeded*
- *sprockets_dynamodb.exceptions.MissingParameter*
- *sprockets_dynamodb.exceptions.OptInRequired*
- *sprockets_dynamodb.exceptions.ResourceInUse*
- *sprockets_dynamodb.exceptions.RequestExpired*
- *sprockets_dynamodb.exceptions.ServiceUnavailable*
- *sprockets_dynamodb.exceptions.ValidationException*

Create an instance of this class to interact with a DynamoDB server. A `tornado_aws.client.AsyncAWSClient` instance implements the AWS API wrapping and this class provides the DynamoDB specifics.

batch_get_item()
Invoke the BatchGetItem function.

batch_write_item()
Invoke the BatchWriteItem function.

create_table(*table_definition*)
Invoke the `CreateTable` function.

Parameters **table_definition** (*dict*) – description of the table to create according to CreateTable

Return type tornado.concurrent.Future

delete_item(*table_name*, *key_dict*, *condition_expression=None*, *expression_attribute_names=None*, *expression_attribute_values=None*, *return_consumed_capacity=None*, *return_item_collection_metrics=None*, *return_values=False*)
Invoke the DeleteItem function that deletes a single item in a table by primary key. You can perform a conditional delete operation that deletes the item if it exists, or if it has an expected attribute value.

Parameters

- **table_name** (*str*) – The name of the table from which to delete the item.
- **key_dict** (*dict*) – A map of attribute names to `AttributeValue` objects, representing the primary key of the item to delete. For the primary key, you must provide all of the attributes. For example, with a simple primary key, you only need to provide a value for the partition key. For a composite primary key, you must provide values for both the partition key and the sort key.

- **condition_expression** (*str*) – A condition that must be satisfied in order for a conditional *DeleteItem* to succeed. See the AWS documentation for ConditionExpression for more information.
- **expression_attribute_names** (*dict*) – One or more substitution tokens for attribute names in an expression. See the AWS documentation for ExpressionAttributeNames for more information.
- **expression_attribute_values** (*dict*) – One or more values that can be substituted in an expression. See the AWS documentation for ExpressionAttributeValues for more information.
- **return_consumed_capacity** (*str*) – Determines the level of detail about provisioned throughput consumption that is returned in the response. See the AWS documentation for ReturnConsumedCapacity for more information.
- **return_item_collection_metrics** (*str*) – Determines whether item collection metrics are returned.
- **return_values** (*str*) – Return the item attributes as they appeared before they were deleted.

delete_table (*table_name*)

Invoke the DeleteTable function. The DeleteTable operation deletes a table and all of its items. After a DeleteTable request, the specified table is in the DELETING state until DynamoDB completes the deletion. If the table is in the ACTIVE state, you can delete it. If a table is in CREATING or UPDATING states, then a *ResourceInUse* exception is raised. If the specified table does not exist, a *ResourceNotFound* exception is raised. If table is already in the DELETING state, no error is returned.

Parameters **table_name** (*str*) – name of the table to describe.

Return type tornado.concurrent.Future

describe_table (*table_name*)

Invoke the DescribeTable function.

Parameters **table_name** (*str*) – name of the table to describe.

Return type tornado.concurrent.Future

execute (**args*, ***kwargs*)

Execute a DynamoDB action with the given parameters. The method will retry requests that failed due to OS level errors or when being throttled by DynamoDB.

Parameters

- **action** (*str*) – DynamoDB action to invoke
- **parameters** (*dict*) – parameters to send into the action

Return type tornado.concurrent.Future

This method creates a future that will resolve to the result of calling the specified DynamoDB function. It does it's best to unwrap the response from the function to make life a little easier for you. It does this for the `GetItem` and `Query` functions currently.

Raises *DynamoDBException* *ConfigNotFound* *NoCredentialsError* *NoProfileError* *TimeoutException* *RequestException* *InternalFailure* *LimitExceeded* *MissingParameter* *OptInRequired* *ResourceInUse* *RequestExpired* *ResourceNotFound* *ServiceUnavailable* *ThroughputExceeded* *ValidationException*

get_item(*table_name*, *key_dict*, *consistent_read=False*, *expression_attribute_names=None*, *projection_expression=None*, *return_consumed_capacity=None*)

Invoke the GetItem function.

Parameters

- **table_name** (*str*) – table to retrieve the item from
- **key_dict** (*dict*) – key to use for retrieval. This will be marshalled for you so a native dict works.
- **consistent_read** (*bool*) – Determines the read consistency model: If set to :py:data‘True‘, then the operation uses strongly consistent reads; otherwise, the operation uses eventually consistent reads.
- **expression_attribute_names** (*dict*) – One or more substitution tokens for attribute names in an expression.
- **projection_expression** (*str*) – A string that identifies one or more attributes to retrieve from the table. These attributes can include scalars, sets, or elements of a JSON document. The attributes in the expression must be separated by commas. If no attribute names are specified, then all attributes will be returned. If any of the requested attributes are not found, they will not appear in the result.
- **return_consumed_capacity** (*str*) – Determines the level of detail about provisioned throughput consumption that is returned in the response:
  - INDEXES: The response includes the aggregate consumed capacity for the operation, together with consumed capacity for each table and secondary index that was accessed. Note that some operations, such as *GetItem* and *BatchGetItem*, do not access any indexes at all. In these cases, specifying INDEXES will only return consumed capacity information for table(s).
  - TOTAL: The response includes only the aggregate consumed capacity for the operation.
  - NONE: No consumed capacity details are included in the response.

Return type tornado.concurrent.Future

list_tables(*exclusive_start_table_name=None*, *limit=None*)

Invoke the ListTables function.

Returns an array of table names associated with the current account and endpoint. The output from *ListTables* is paginated, with each page returning a maximum of `100` table names.

Parameters

- **exclusive_start_table_name** (*str*) – The first table name that this operation will evaluate. Use the value that was returned for `LastEvaluatedTableName` in a previous operation, so that you can obtain the next page of results.
- **limit** (*int*) – A maximum number of table names to return. If this parameter is not specified, the limit is `100`.

put_item(*table_name*, *item*, *condition_expression=None*, *expression_attribute_names=None*, *expression_attribute_values=None*, *return_consumed_capacity=None*, *return_item_collection_metrics=None*, *return_values=None*)

Invoke the PutItem function, creating a new item, or replaces an old item with a new item. If an item that has the same primary key as the new item already exists in the specified table, the new item completely replaces the existing item. You can perform a conditional put operation (add a new item if one with the specified primary key doesn’t exist), or replace an existing item if it has certain attribute values.

For more information about using this API, see Working with Items in the Amazon DynamoDB Developer Guide.

Parameters

- **`table_name`** (*`str`*) – The table to put the item to
- **`item`** (*`dict`*) – A map of attribute name/value pairs, one for each attribute. Only the primary key attributes are required; you can optionally provide other attribute name-value pairs for the item.

  You must provide all of the attributes for the primary key. For example, with a simple primary key, you only need to provide a value for the partition key. For a composite primary key, you must provide both values for both the partition key and the sort key.

  If you specify any attributes that are part of an index key, then the data types for those attributes must match those of the schema in the table's attribute definition.
- **`condition_expression`** (*`str`*) – A condition that must be satisfied in order for a conditional *PutItem* operation to succeed. See the AWS documentation for ConditionExpression for more information.
- **`expression_attribute_names`** (*`dict`*) – One or more substitution tokens for attribute names in an expression. See the AWS documentation for ExpressionAttributeNames for more information.
- **`expression_attribute_values`** (*`dict`*) – One or more values that can be substituted in an expression. See the AWS documentation for ExpressionAttributeValues for more information.
- **`return_consumed_capacity`** (*`str`*) – Determines the level of detail about provisioned throughput consumption that is returned in the response. Should be `None` or one of `INDEXES` or `TOTAL`
- **`return_item_collection_metrics`** (*`str`*) – Determines whether item collection metrics are returned.
- **`return_values`** (*`str`*) – Use `ReturnValues` if you want to get the item attributes as they appeared before they were updated with the `PutItem` request.

Return type tornado.concurrent.Future

`query`(*table_name*, *index_name=None*, *consistent_read=None*, *key_condition_expression=None*, *filter_expression=None*, *expression_attribute_names=None*, *expression_attribute_values=None*, *projection_expression=None*, *select=None*, *exclusive_start_key=None*, *limit=None*, *scan_index_forward=True*, *return_consumed_capacity=None*)

A Query operation uses the primary key of a table or a secondary index to directly access items from that table or index.

Parameters

- **`table_name`** (*`str`*) – The name of the table containing the requested items.
- **`consistent_read`** (*`bool`*) – Determines the read consistency model: If set to `True`, then the operation uses strongly consistent reads; otherwise, the operation uses eventually consistent reads. Strongly consistent reads are not supported on global secondary indexes. If you query a global secondary index with `consistent_read` set to `True`, you will receive a *`ValidationException`*.
- **`exclusive_start_key`** (*`dict`*) – The primary key of the first item that this operation will evaluate. Use the value that was returned for `LastEvaluatedKey` in the previous operation. In a parallel scan, a *Scan* request that includes `exclusive_start_key`

must specify the same segment whose previous *Scan* returned the corresponding value of `LastEvaluatedKey`.

- **`expression_attribute_names`** (*dict*) – One or more substitution tokens for attribute names in an expression.
- **`expression_attribute_values`** (*dict*) – One or more values that can be substituted in an expression.
- **`filter_expression`** (*str*) – A string that contains conditions that DynamoDB applies after the *Query* operation, but before the data is returned to you. Items that do not satisfy the criteria are not returned. Note that a filter expression is applied after the items have already been read; the process of filtering does not consume any additional read capacity units. For more information, see Filter Expressions in the Amazon DynamoDB Developer Guide.
- **`projection_expression`** (*str*) –
- **`index_name`** (*str*) – The name of a secondary index to query. This index can be any local secondary index or global secondary index. Note that if you use this parameter, you must also provide `table_name`.
- **`limit`** (*int*) – The maximum number of items to evaluate (not necessarily the number of matching items). If DynamoDB processes the number of items up to the limit while processing the results, it stops the operation and returns the matching values up to that point, and a key in `LastEvaluatedKey` to apply in a subsequent operation, so that you can pick up where you left off. Also, if the processed data set size exceeds 1 MB before DynamoDB reaches this limit, it stops the operation and returns the matching values up to the limit, and a key in `LastEvaluatedKey` to apply in a subsequent operation to continue the operation. For more information, see Query and Scan in the Amazon DynamoDB Developer Guide.
- **`return_consumed_capacity`** (*str*) – Determines the level of detail about provisioned throughput consumption that is returned in the response:
  - `INDEXES`: The response includes the aggregate consumed capacity for the operation, together with consumed capacity for each table and secondary index that was accessed. Note that some operations, such as *GetItem* and *BatchGetItem*, do not access any indexes at all. In these cases, specifying `INDEXES` will only return consumed capacity information for table(s).
  - `TOTAL`: The response includes only the aggregate consumed capacity for the operation.
  - `NONE`: No consumed capacity details are included in the response.
- **`scan_index_forward`** (*bool*) – Specifies the order for index traversal: If `True` (default), the traversal is performed in ascending order; if `False`, the traversal is performed in descending order. Items with the same partition key value are stored in sorted order by sort key. If the sort key data type is *Number*, the results are stored in numeric order. For type *String*, the results are stored in order of ASCII character code values. For type *Binary*, DynamoDB treats each byte of the binary data as unsigned. If set to `True`, DynamoDB returns the results in the order in which they are stored (by sort key value). This is the default behavior. If set to `False`, DynamoDB reads the results in reverse order by sort key value, and then returns the results to the client.
- **`select`** (*str*) – The attributes to be returned in the result. You can retrieve all item attributes, specific item attributes, the count of matching items, or in the case of an index, some or all of the attributes projected into the index. Possible values are:

- `ALL_ATTRIBUTES`: Returns all of the item attributes from the specified table or index. If you query a local secondary index, then for each matching item in the index DynamoDB will fetch the entire item from the parent table. If the index is configured to project all item attributes, then all of the data can be obtained from the local secondary index, and no fetching is required.
- `ALL_PROJECTED_ATTRIBUTES`: Allowed only when querying an index. Retrieves all attributes that have been projected into the index. If the index is configured to project all attributes, this return value is equivalent to specifying `ALL_ATTRIBUTES`.
- `COUNT`: Returns the number of matching items, rather than the matching items themselves.

Return type dict

`scan`(*table_name*, *index_name=None*, *consistent_read=None*, *projection_expression=None*, *filter_expression=None*, *expression_attribute_names=None*, *expression_attribute_values=None*, *segment=None*, *total_segments=None*, *select=None*, *limit=None*, *exclusive_start_key=None*, *return_consumed_capacity=None*)

The Scan operation returns one or more items and item attributes by accessing every item in a table or a secondary index.

If the total number of scanned items exceeds the maximum data set size limit of 1 MB, the scan stops and results are returned to the user as a `LastEvaluatedKey` value to continue the scan in a subsequent operation. The results also include the number of items exceeding the limit. A scan can result in no table data meeting the filter criteria.

By default, Scan operations proceed sequentially; however, for faster performance on a large table or secondary index, applications can request a parallel *Scan* operation by providing the `segment` and `total_segments` parameters. For more information, see Parallel Scan in the Amazon DynamoDB Developer Guide.

By default, *Scan* uses eventually consistent reads when accessing the data in a table; therefore, the result set might not include the changes to data in the table immediately before the operation began. If you need a consistent copy of the data, as of the time that the *Scan* begins, you can set the `consistent_read` parameter to `True`.

Return type dict

`set_error_callback`(*callback*)

Assign a method to invoke when a request has encountered an unrecoverable error in an action execution.

Parameters **`callback`** (*`method`*) – The method to invoke

`set_instrumentation_callback`(*callback*)

Assign a method to invoke when a request has completed gathering measurements.

Parameters **`callback`** (*`method`*) – The method to invoke

`update_item`(*table_name*, *key_dict*, *condition_expression=None*, *update_expression=None*, *expression_attribute_names=None*, *expression_attribute_values=None*, *return_consumed_capacity=None*, *return_item_collection_metrics=None*, *return_values=None*)

Invoke the UpdateItem function.

Edits an existing item's attributes, or adds a new item to the table if it does not already exist. You can put, delete, or add attribute values. You can also perform a conditional update on an existing item (insert a new attribute name-value pair if it doesn't exist, or replace an existing name-value pair if it has certain expected attribute values).

Parameters

- **`table_name`** (*`str`*) – The name of the table that contains the item to update
- **`key_dict`** (*`dict`*) – A dictionary of key/value pairs that are used to define the primary key values for the item. For the primary key, you must provide all of the attributes. For example, with a simple primary key, you only need to provide a value for the partition key. For a composite primary key, you must provide values for both the partition key and the sort key.
- **`condition_expression`** (*`str`*) – A condition that must be satisfied in order for a conditional *UpdateItem* operation to succeed. One of: `attribute_exists`, `attribute_not_exists`, `attribute_type`, `contains`, `begins_with`, `size`, `=`, `<>`, `<`, `>`, `<=`, `>=`, `BETWEEN`, `IN`, `AND`, `OR`, or `NOT`.
- **`update_expression`** (*`str`*) – An expression that defines one or more attributes to be updated, the action to be performed on them, and new value(s) for them.
- **`expression_attribute_names`** (*`dict`*) – One or more substitution tokens for attribute names in an expression.
- **`expression_attribute_values`** (*`dict`*) – One or more values that can be substituted in an expression.
- **`return_consumed_capacity`** (*`str`*) – Determines the level of detail about provisioned throughput consumption that is returned in the response. See the AWS documentation for ReturnConsumedCapacity for more information.
- **`return_item_collection_metrics`** (*`str`*) – Determines whether item collection metrics are returned.
- **`return_values`** (*`str`*) – Use ReturnValues if you want to get the item attributes as they appeared either before or after they were updated. See the AWS documentation for ReturnValues

Return type tornado.concurrent.Future

`update_table`(*table_definition*)

Modifies the provisioned throughput settings, global secondary indexes, or DynamoDB Streams settings for a given table.

You can only perform one of the following operations at once:

- Modify the provisioned throughput settings of the table.
- Enable or disable Streams on the table.
- Remove a global secondary index from the table.
- Create a new global secondary index on the table. Once the index begins back-filling, you can use *UpdateTable* to perform other operations.

UpdateTable is an asynchronous operation; while it is executing, the table status changes from `ACTIVE` to `UPDATING`. While it is `UPDATING`, you cannot issue another *UpdateTable* request. When the table returns to the `ACTIVE` state, the *UpdateTable* operation is complete.

Parameters **`table_definition`** (*`dict`*) – description of the table to update according to UpdateTable

Return type tornado.concurrent.Future

5.2 DynamoDB Exceptions

exception `sprockets_dynamodb.exceptions.`**`ConditionalCheckFailedException`**(**args*, ***kwargs*)

A condition specified in the operation could not be evaluated.

exception `sprockets_dynamodb.exceptions.`**`ConfigNotFound`**(**args*, ***kwargs*)

The configuration file could not be parsed.

exception `sprockets_dynamodb.exceptions.`**`ConfigParserError`**(**args*, ***kwargs*)

Error raised when parsing a configuration file with `RawConfigParser`

exception `sprockets_dynamodb.exceptions.`**`DynamoDBException`**(**args*, ***kwargs*)

Base exception that is extended by all exceptions raised by tornado_dynamodb.

Variables **`msg`** – The error message

exception `sprockets_dynamodb.exceptions.`**`InternalFailure`**(**args*, ***kwargs*)

The request processing has failed because of an unknown error, exception or failure.

exception `sprockets_dynamodb.exceptions.`**`InternalServerError`**(**args*, ***kwargs*)

The request processing has failed because DynamoDB could not process your request.

exception `sprockets_dynamodb.exceptions.`**`InvalidAction`**(**args*, ***kwargs*)

The action or operation requested is invalid. Verify that the action is typed correctly.

exception `sprockets_dynamodb.exceptions.`**`InvalidParameterCombination`**(**args*, ***kwargs*)

Parameters that must not be used together were used together.

exception `sprockets_dynamodb.exceptions.`**`InvalidParameterValue`**(**args*, ***kwargs*)

An invalid or out-of-range value was supplied for the input parameter.

exception `sprockets_dynamodb.exceptions.`**`InvalidQueryParameter`**(**args*, ***kwargs*)

The AWS query string is malformed or does not adhere to AWS standards.

exception `sprockets_dynamodb.exceptions.`**`ItemCollectionSizeLimitExceeded`**(**args*, ***kwargs*)

An item collection is too large. This exception is only returned for tables that have one or more local secondary indexes.

exception `sprockets_dynamodb.exceptions.`**`LimitExceeded`**(**args*, ***kwargs*)

The number of concurrent table requests (cumulative number of tables in the `CREATING`, `DELETING` or `UPDATING` state) exceeds the maximum allowed of `10`.

Also, for tables with secondary indexes, only one of those tables can be in the `CREATING` state at any point in time. Do not attempt to create more than one such table simultaneously.

The total limit of tables in the `ACTIVE` state is `250`.

exception `sprockets_dynamodb.exceptions.`**`MalformedQueryString`**(**args*, ***kwargs*)

The query string contains a syntax error.

exception `sprockets_dynamodb.exceptions.`**`MissingParameter`**(**args*, ***kwargs*)

A required parameter for the specified action is not supplied.

exception `sprockets_dynamodb.exceptions.`**`NoCredentialsError`**(**args*, ***kwargs*)

Raised when the credentials could not be located.

exception `sprockets_dynamodb.exceptions.`**`NoProfileError`**(**args*, ***kwargs*)

Raised when the specified profile could not be located.

exception sprockets_dynamodb.exceptions.**OptInRequired**(**args*, ***kwargs*)
The AWS access key ID needs a subscription for the service.

exception sprockets_dynamodb.exceptions.**RequestException**(**args*, ***kwargs*)
Raised when the HTTP request failed due to a network or DNS related issue.

exception sprockets_dynamodb.exceptions.**RequestExpired**(**args*, ***kwargs*)
The request reached the service more than 15 minutes after the date stamp on the request or more than 15 minutes after the request expiration date (such as for pre-signed URLs), or the date stamp on the request is more than 15 minutes in the future.

exception sprockets_dynamodb.exceptions.**ResourceInUse**(**args*, ***kwargs*)
he operation conflicts with the resource's availability. For example, you attempted to recreate an existing table, or tried to delete a table currently in the `CREATING` state.

exception sprockets_dynamodb.exceptions.**ResourceNotFound**(**args*, ***kwargs*)
The operation tried to access a nonexistent table or index. The resource might not be specified correctly, or its status might not be `ACTIVE`.

exception sprockets_dynamodb.exceptions.**ServiceUnavailable**(**args*, ***kwargs*)
The request has failed due to a temporary failure of the server.

exception sprockets_dynamodb.exceptions.**ThrottlingException**(**args*, ***kwargs*)
This exception might be returned if the following API operations are requested too rapidly: CreateTable; UpdateTable; DeleteTable.

exception sprockets_dynamodb.exceptions.**ThroughputExceeded**(**args*, ***kwargs*)
Your request rate is too high. The AWS SDKs for DynamoDB automatically retry requests that receive this exception. Your request is eventually successful, unless your retry queue is too large to finish. Reduce the frequency of requests and use exponential backoff. For more information, go to Error Retries and Exponential Backoff in the Amazon DynamoDB Developer Guide.

exception sprockets_dynamodb.exceptions.**TimeoutException**(**args*, ***kwargs*)
The request to DynamoDB timed out.

exception sprockets_dynamodb.exceptions.**ValidationException**(**args*, ***kwargs*)
The input fails to satisfy the constraints specified by an AWS service. This error can occur for several reasons, such as a required parameter that is missing, a value that is out range, or mismatched data types. The error message contains details about the specific part of the request that caused the error.

5.3 Examples

5.3.1 Creating a Table

```
"""
Create the table described in `CreateTable`_.

This example creates a table if it does not exist using chained
callbacks.

.. _CreateTable: http://docs.aws.amazon.com/amazondynamodb/latest/
   APIReference/API_CreateTable.html

"""
import logging
import sys
```

```python
import sprockets_dynamodb as dynamodb
from tornado import ioloop


LOGGER = logging.getLogger('create-database')
TABLE_DEF = {
    'TableName': 'Thread',
    'AttributeDefinitions': [
        {'AttributeName': 'ForumName', 'AttributeType': 'S'},
        {'AttributeName': 'Subject', 'AttributeType': 'S'},
        {'AttributeName': 'LastPostDateTime', 'AttributeType': 'S'},
    ],
    'KeySchema': [
        {'AttributeName': 'ForumName', 'KeyType': 'HASH'},
        {'AttributeName': 'Subject', 'KeyType': 'RANGE'},
    ],
    'LocalSecondaryIndexes': [
        {
            'IndexName': 'LastPostIndex',
            'KeySchema': [
                {'AttributeName': 'ForumName', 'KeyType': 'HASH'},
                {'AttributeName': 'LastPostDateTime', 'KeyType': 'RANGE'},
            ],
            'Projection': {
                'ProjectionType': 'KEYS_ONLY',
            },
        },
    ],
    'ProvisionedThroughput': {
        'ReadCapacityUnits': 5,
        'WriteCapacityUnits': 5,
    }
}

dynamo = dynamodb.Client()
iol = ioloop.IOLoop.instance()


def on_table_described(describe_response):

    def on_created(create_response):
        try:
            result = create_response.result()
            LOGGER.info('table created - %r', result)
            iol.add_callback(iol.stop)
        except Exception:
            LOGGER.exception('failed to create table')
            sys.exit(-1)

    try:
        result = describe_response.result()
        LOGGER.info('found table %s, created %s',
                    result['Table']['TableName'],
                    result['Table']['CreationDateTime'])
        iol.add_callback(iol.stop)
    except Exception as error:
        LOGGER.warn('table not found, attempting to create: %s', error)
        next_future = dynamo.create_table(TABLE_DEF)
        iol.add_future(next_future, on_created)

logging.basicConfig(level=logging.DEBUG,
                    format='%(levelname)1.1s %(name)s: %(message)s')

iol.add_future(dynamo.describe_table(TABLE_DEF['TableName']),
               on_table_described)
iol.start()
```

5.4 How to Contribute

Do you want to contribute fixes or improvements?

> **AWesome!** *Thank you very much, and let's get started.*

5.4.1 Set up a development environment

The first thing that you need is a development environment so that you can run the test suite, update the documentation, and everything else that is involved in contributing. The easiest way to do that is to create a virtual environment for your endeavours:

```
$ python3 -mvenv env
```

Don't worry about writing code against previous versions of Python unless you you don't have a choice. That is why we run our tests through tox. If you don't have a choice, then install virtualenv to create the environment instead. The next step is to install the development tools that this project uses. These are listed in *requires/development.txt*:

```
$ env/bin/pip install -qr requires/development.txt
```

At this point, you will have everything that you need to develop at your disposal. *setup.py* is the swiss-army knife in your development tool chest. It provides the following commands:

./setup.py nosetests Run the test suite using nose and generate a nice coverage report.

./setup.py build_sphinx Generate the documentation using sphinx.

./setup.py flake8 Run flake8 over the code and report style violations.

If any of the preceding commands give you problems, then you will have to fix them **before** your pull request will be accepted.

5.4.2 Running Tests

The easiest (and quickest) way to run the test suite is to use the *nosetests* command. It will run the test suite against the currently installed python version and report not only the test result but the test coverage as well:

```
$ ./setup.py nosetests

running nosetests
running egg_info
writing dependency_links to sprockets_dynamodb/dependency_links.txt
writing top-level names to sprockets_dynamodb/top_level.txt
writing sprockets_dynamodb/PKG-INFO
reading manifest file 'sprockets_dynamodb/SOURCES.txt'
reading manifest template 'MANIFEST.in'
warning: no previously-included files matching '__pycache__'...
warning: no previously-included files matching '*.swp' found ...
writing manifest file 'sprockets_dynamodb/SOURCES.txt'
...

Name                           Stmts   Miss Branch BrMiss  Cover   Missing
----------------------------------------------------------------------
...
----------------------------------------------------------------------
TOTAL                             95      2     59      2    97%
----------------------------------------------------------------------
Ran 44 tests in 0.054s

OK
```

That's the quick way to run tests. The slightly longer way is to run the tox utility. It will run the test suite against all of the supported python versions in parallel. This is essentially what Travis-CI will do when you issue a pull request anyway:

```
$ env/bin/tox
py27 recreate: /.../sprockets-dynamodb/build/tox/py27
GLOB sdist-make: /.../sprockets-dynamodb/setup.py
py34 recreate: /.../sprockets-dynamodb/build/tox/py34
py27 installdeps: -rtest-requirements.txt, mock
py34 installdeps: -rtest-requirements.txt
py27 inst: /.../sprockets-dynamodb/build/tox/dist/sprockets-dynamodb-0.0.0.zip
py27 runtests: PYTHONHASHSEED='2156646470'
py27 runtests: commands[0] | /../sprockets-dynamodb/build/tox/py27/bin/nosetests
py34 inst: /../sprockets-dynamodb/.build/tox/dist/sprockets-dynamodb-0.0.0.zip
py34 runtests: PYTHONHASHSEED='2156646470'
py34 runtests: commands[0] | /.../sprockets-dynamodb/build/tox/py34/bin/nosetests
_________________________________ summary _________________________________
  py27: commands succeeded
  py34: commands succeeded
  congratulations :)
```

This is what you want to see. Now you can make your modifications and keep the tests passing.

5.4.3 Submitting a Pull Request

Once you have made your modifications, gotten all of the tests to pass, and added any necessary documentation, it is time to contribute back for posterity. You've probably already cloned this repository and created a new branch. If you haven't, then checkout what you have as a branch and roll back *master* to where you found it. Then push your repository up to github and issue a pull request. Describe your changes in the request, if Travis isn't too annoyed someone will review it, and eventually merge it back.

5.5 Release History

5.5.1 2.3.0 (24 Jan 2018)

- Add handling for `tornado_aws.exceptions.RequestException`
- Update to tornado-aws 1.0.0

5.5.2 2.2.0 (06 Jun 2017)

- Update to tornado-aws 0.8.0 and add Python 3.6 to supported versions

5.5.3 2.1.3 (27 Jan 2017)

- Add HTTP Internal Server Error (500) and Service Unavailable (503) to retriable exceptions

5.5.4 2.1.2 (28 Oct 2016)

- Change the pinning of tornado-aws to open it up a little wider

5.5.5 2.1.1 (21 Oct 2016)

- Add `DYANMODB_NO_CREDS_RATE_LIMIT` environment variable support to the mixin

5.5.6 2.1.0 (20 Oct 2016)

- Fix exception handling for requests to actually catch all the exceptions we care about

5.5.7 2.0.0 (17 Oct 2016)

- Breaking API change for Client.get_item to allow for return values for ConsumedCapacity
- Implement Client.delete_item, Client.update_item, Client.query, Client.scan
- Improved parameter validation

5.5.8 1.1.0 (12 Oct 2016)

- Remove the service tag in InfluxDB
- Remove the correlation-id field value
- Collect all of the paged query results

5.5.9 1.0.2 (26 Sep 2016)

- Fix a mixin InfluxDB integration issue

5.5.10 1.0.1 (26 Sep 2016)

- Make `DynamoDBMixin` available from `sprockets_dynamodb`

5.5.11 1.0.0 (26 Sep 2016)

- Initial release

5.5.12 Next Release

Python Module Index

Index