tornado-dynamodb

Release 0.1.0

Mar 17, 2017

Contents

`tornado-dynamodb` is an Asynchronous DynamoDB client Tornado

Contents:

CHAPTER 1

API

DynamoDB Client

DynamoDB is an opinionated DynamoDB client for Tornado. While it follows the DynamoDB API fairly closely, it does try and make some of the more mundane tasks like data marshalling and demarshalling for you.

class tornado_dynamodb.**DynamoDB**(*profile=None*, *region=None*, *access_key=None*, *secret_key=None*, *endpoint=None*, *max_clients=100*)

An opinionated asynchronous DynamoDB client for Tornado

Parameters

- **profile** (*str*) – Specify the configuration profile name
- **region** (*str*) – The AWS region to make requests to
- **access_key** (*str*) – The access key
- **secret_key** (*str*) – The secret access key
- **endpoint** (*str*) – Override the base endpoint URL
- **max_clients** (*int*) – Max simultaneous requests (Default: 100)

Raises *ConfigNotFound* *ConfigParserError* *NoCredentialsError* *NoProfileError*

create_table(*name*, *attributes*, *key_schema*, *read_capacity_units=1*, *write_capacity_units=1*, *global_secondary_indexes=None*, *local_secondary_indexes=None*, *stream_enabled=False*, *stream_view_type=None*)

The *CreateTable* operation adds a new table to your account. In an AWS account, table names must be unique within each region. That is, you can have two tables with same name if you create the tables in different regions.

CreateTable is an asynchronous operation. Upon receiving a *CreateTable* request, DynamoDB immediately returns a response with a TableStatus of CREATING. After the table is created, DynamoDB sets the TableStatus to ACTIVE. You can perform read and write operations only on an ACTIVE table.

You can optionally define secondary indexes on the new table, as part of the CreateTable operation. If you want to create multiple tables with secondary indexes on them, you must create the tables sequentially. Only one table with secondary indexes can be in the `CREATING` state at any given time.

For the proper format of `attributes`, `key_schema`, `local_secondary_indexes`, and `global_secondary_indexes` please visit the Amazon documentation for the CreateTable operation.

You can use the *`describe_table()`* API to check the table status.

Parameters

- **`name`** (*`str`*) – The table name
- **`attributes`** (*`list`*) – A list of attribute definition key/value pairs where the key is the name of the attribute and the value is one of `S`, `N`, or `B` indicating the data type of the attribute.
- **`key_schema`** (*`list`*) – A list of key definitions that specify the attributes that make up the primary key for a table or an index. Each key pair in the list consists of the attribute name as the key and the index type as the value.
- **`read_capacity_units`** (*`int`*) – The maximum number of strongly consistent reads consumed per second before DynamoDB returns a *`ThrottlingException`*
- **`write_capacity_units`** (*`int`*) – The maximum number of writes consumed per second before DynamoDB returns a *`ThrottlingException`*
- **`global_secondary_indexes`** (*`list`*) – One or more global secondary indexes (the maximum is five) to be created on the table.
- **`local_secondary_indexes`** (*`list`*) – One or more local secondary indexes (the maximum is five) to be created on the table. Each index is scoped to a given partition key value. There is a 10 GB size limit per partition key value; otherwise, the size of a local secondary index is unconstrained.
- **`stream_enabled`** (*`bool`*) – Indicates whether DynamoDB Streams is enabled (`True`) or disabled (`False`) for the table.
- **`stream_view_type`** (*`str`*) – When an item in the table is modified, `stream_view_type` determines what information is written to the stream for this table.

Returns

Response format:

```
{
  "AttributeDefinitions": [{
    "AttributeName": "string",
    "AttributeType": "string"
  }],
  "CreationDateTime": number,
  "GlobalSecondaryIndexes": [{
    "Backfilling": boolean,
    "IndexArn": "string",
    "IndexName": "string",
    "IndexSizeBytes": number,
    "IndexStatus": "string",
    "ItemCount": number,
    "KeySchema": [{
      "AttributeName": "string",
```

```
      "KeyType": "string"
    }],
    "Projection": {
      "NonKeyAttributes": [
        "string"
      ],
      "ProjectionType": "string"
    },
    "ProvisionedThroughput": {
      "LastDecreaseDateTime": number,
      "LastIncreaseDateTime": number,
      "NumberOfDecreasesToday": number,
      "ReadCapacityUnits": number,
      "WriteCapacityUnits": number
    }
  }],
  "ItemCount": number,
  "KeySchema": [{
    "AttributeName": "string",
    "KeyType": "string"
  }],
  "LatestStreamArn": "string",
  "LatestStreamLabel": "string",
  "LocalSecondaryIndexes": [{
    "IndexArn": "string",
    "IndexName": "string",
    "IndexSizeBytes": number,
    "ItemCount": number,
    "KeySchema": [{
      "AttributeName": "string",
      "KeyType": "string"
    }],
    "Projection": {
      "NonKeyAttributes": [
        "string"
      ],
      "ProjectionType": "string"
    }
  }],
  "ProvisionedThroughput": {
    "LastDecreaseDateTime": number,
    "LastIncreaseDateTime": number,
    "NumberOfDecreasesToday": number,
    "ReadCapacityUnits": number,
    "WriteCapacityUnits": number
  },
  "StreamSpecification": {
    "StreamEnabled": boolean,
    "StreamViewType": "string"
  },
  "TableArn": "string",
  "TableName": "string",
  "TableSizeBytes": number,
  "TableStatus": "string"
}
```

Raises *InternalFailure* *LimitExceeded* *MissingParameter* *OptInRequired* *ResourceInUse* *RequestExpired* *ServiceUnavailable*

ThrottlingException *ValidationException*

delete_item(*table_name*, *key*, *condition_expression=None*, *expression_attribute_names=None*, *expression_attribute_values=None*, *return_consumed_capacity=None*, *return_item_collection_metrics=False*, *return_values=False*)

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.

In addition to deleting an item, you can also return the item's attribute values in the same operation, using the `return_values` parameter.

Unless you specify conditions, the *DeleteItem* is an idempotent operation; running it multiple times on the same item or attribute does not result in an error response.

Conditional deletes are useful for deleting items only if specific conditions are met. If those conditions are met, DynamoDB performs the delete. Otherwise, the item is not deleted.

Parameters

- **table_name** (*str*) – The name of the table from which to delete the item.
- **key** (*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** (*bool*) – Determines whether item collection metrics are returned.
- **return_values** (*bool*) – Return the item attributes as they appeared before they were deleted.

Returns

Response format:

```
{
  "Attributes": {
    "string": {
      "B": blob,
      "BOOL": boolean,
      "BS": [
        blob
      ],
      "L": [
```

```
        AttributeValue
      ],
      "M": {
        "string": AttributeValue
      },
      "N": "string",
      "NS": [
        "string"
      ],
      "NULL": boolean,
      "S": "string",
      "SS": [
        "string"
      ]
    }
  },
  "ConsumedCapacity": {
    "CapacityUnits": number,
    "GlobalSecondaryIndexes": {
      "string": {
        "CapacityUnits": number
      }
    },
    "LocalSecondaryIndexes": {
      "string": {
        "CapacityUnits": number
      }
    },
    "Table": {
      "CapacityUnits": number
    },
    "TableName": "string"
  },
  "ItemCollectionMetrics": {
    "ItemCollectionKey": {
      "string": {
        "B": blob,
        "BOOL": boolean,
        "BS": [
          blob
        ],
        "L": [
          AttributeValue
        ],
        "M": {
          "string": AttributeValue
        },
        "N": "string",
        "NS": [
          "string"
        ],
        "NULL": boolean,
        "S": "string",
        "SS": [
          "string"
        ]
      }
    },
```

```
    "SizeEstimateRangeGB": [
      number
    ]
  }
}
```

Raises *InternalFailure* *MissingParameter* *OptInRequired* *RequestExpired* *ServiceUnavailable* *ThrottlingException* *ValidationException* *ResourceNotFound* *ProvisionedThroughputExceeded* *ItemCollectionSizeLimitExceeded*

delete_table(*name*)

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 DynamoDB returns a *ResourceInUse*. If the specified table does not exist, DynamoDB returns a *ResourceNotFound* . If table is already in the DELETING state, no error is returned.

Parameters **name** (*str*) – The table name

Returns

Response Format:

```
{
  "AttributeDefinitions": [{
    "AttributeName": "string",
    "AttributeType": "string"
  }],
  "CreationDateTime": number,
  "GlobalSecondaryIndexes": [{
    "Backfilling": boolean,
    "IndexArn": "string",
    "IndexName": "string",
    "IndexSizeBytes": number,
    "IndexStatus": "string",
    "ItemCount": number,
    "KeySchema": [{
      "AttributeName": "string",
      "KeyType": "string"
    }],
    "Projection": {
      "NonKeyAttributes": [
        "string"
      ],
      "ProjectionType": "string"
    },
    "ProvisionedThroughput": {
      "LastDecreaseDateTime": number,
      "LastIncreaseDateTime": number,
      "NumberOfDecreasesToday": number,
      "ReadCapacityUnits": number,
      "WriteCapacityUnits": number
    }
  }],
  "ItemCount": number,
  "KeySchema": [{
    "AttributeName": "string",
    "KeyType": "string"
```

```
  }],
  "LatestStreamArn": "string",
  "LatestStreamLabel": "string",
  "LocalSecondaryIndexes": [{
    "IndexArn": "string",
    "IndexName": "string",
    "IndexSizeBytes": number,
    "ItemCount": number,
    "KeySchema": [{
      "AttributeName": "string",
      "KeyType": "string"
    }],
    "Projection": {
      "NonKeyAttributes": [
        "string"
      ],
      "ProjectionType": "string"
    }
  }],
  "ProvisionedThroughput": {
    "LastDecreaseDateTime": number,
    "LastIncreaseDateTime": number,
    "NumberOfDecreasesToday": number,
    "ReadCapacityUnits": number,
    "WriteCapacityUnits": number
  },
  "StreamSpecification": {
    "StreamEnabled": boolean,
    "StreamViewType": "string"
  },
  "TableArn": "string",
  "TableName": "string",
  "TableSizeBytes": number,
  "TableStatus": "string"
}
```

Raises *InternalFailure* *MissingParameter* *OptInRequired* *RequestExpired* *ServiceUnavailable* *ThrottlingException* *ValidationException* *LimitExceeded* *ResourceInUse* *ResourceNotFound*

describe_table(*name*)

Returns information about the table, including the current status of the table, when it was created, the primary key schema, and any indexes on the table.

Parameters **name** (*str*) – The table name

Returns

Response Format:

```
{
  "AttributeDefinitions": [{
    "AttributeName": "string",
    "AttributeType": "string"
  }],
  "CreationDateTime": number,
  "GlobalSecondaryIndexes": [{
    "Backfilling": boolean,
```

```
    "IndexArn": "string",
    "IndexName": "string",
    "IndexSizeBytes": number,
    "IndexStatus": "string",
    "ItemCount": number,
    "KeySchema": [{
      "AttributeName": "string",
      "KeyType": "string"
    }],
    "Projection": {
      "NonKeyAttributes": [
        "string"
      ],
      "ProjectionType": "string"
    },
    "ProvisionedThroughput": {
      "LastDecreaseDateTime": number,
      "LastIncreaseDateTime": number,
      "NumberOfDecreasesToday": number,
      "ReadCapacityUnits": number,
      "WriteCapacityUnits": number
    }
  }],
  "ItemCount": number,
  "KeySchema": [{
    "AttributeName": "string",
    "KeyType": "string"
  }],
  "LatestStreamArn": "string",
  "LatestStreamLabel": "string",
  "LocalSecondaryIndexes": [{
    "IndexArn": "string",
    "IndexName": "string",
    "IndexSizeBytes": number,
    "ItemCount": number,
    "KeySchema": [{
      "AttributeName": "string",
      "KeyType": "string"
    }],
    "Projection": {
      "NonKeyAttributes": [
        "string"
      ],
      "ProjectionType": "string"
    }
  }],
  "ProvisionedThroughput": {
    "LastDecreaseDateTime": number,
    "LastIncreaseDateTime": number,
    "NumberOfDecreasesToday": number,
    "ReadCapacityUnits": number,
    "WriteCapacityUnits": number
  },
  "StreamSpecification": {
    "StreamEnabled": boolean,
    "StreamViewType": "string"
  },
  "TableArn": "string",
```

```
  "TableName": "string",
  "TableSizeBytes": number,
  "TableStatus": "string"
}
```

Raises *InternalFailure* *MissingParameter* *OptInRequired* *RequestExpired* *ServiceUnavailable* *ThrottlingException* *ValidationException* *ResourceNotFound*

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

The *GetItem* operation returns a set of attributes for the item with the given primary key. If there is no matching item, *GetItem* does not return any data.

GetItem provides an eventually consistent read by default. If your application requires a strongly consistent read, set `consistent_read` to true. Although a strongly consistent read might take more time than an eventually consistent read, it always returns the last updated value.

Parameters

- **table_name** (*str*) – The name of the table containing the requested item.
- **key** (*dict*) – A map of attribute names to `AttributeValue` objects, representing the primary key of the item to retrieve. 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.
- **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.

Returns

Response Format:

```
{
  "ConsumedCapacity": {
    "CapacityUnits": number,
```

```
    "GlobalSecondaryIndexes": {
      "string": {
        "CapacityUnits": number
      }
    },
    "LocalSecondaryIndexes": {
      "string": {
        "CapacityUnits": number
      }
    },
    "Table": {
      "CapacityUnits": number
    },
    "TableName": "string"
  },
  "Item": {
    "string": {
      "B": blob,
      "BOOL": boolean,
      "BS": [
        blob
      ],
      "L": [
        AttributeValue
      ],
      "M": {
        "string": AttributeValue
      },
      "N": "string",
      "NS": [
        "string"
      ],
      "NULL": boolean,
      "S": "string",
      "SS": [
        "string"
      ]
    }
  }
}
```

Raises *InternalFailure* *MissingParameter* *OptInRequired* *RequestExpired* *ServiceUnavailable* *ThrottlingException* *ValidationException* *ResourceNotFound* *ProvisionedThroughputExceeded*

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

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`.

Returns

Response Format:

```
{
  "LastEvaluatedTableName": "string",
  "TableNames": [
    "string"
  ]
}
```

Raises *InternalFailure* *MissingParameter* *OptInRequired* *RequestExpired* *ServiceUnavailable* *ThrottlingException* *ValidationException*

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

Creates 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.

In addition to putting an item, you can also return the item's attribute values in the same operation, using the `return_values` parameter.

When you add an item, the primary key attribute(s) are the only required attributes. Attribute values cannot be null. String and Binary type attributes must have lengths greater than zero. Set type attributes cannot be empty. Requests with empty values will be rejected with a *ValidationException*.

You can request that PutItem return either a copy of the original item (before the update) or a copy of the updated item (after the update). For more information, see the ReturnValues description below.

Note: To prevent a new item from replacing an existing item, use a conditional expression that contains the attribute_not_exists function with the name of the attribute being used as the partition key for the table. Since every record must contain that attribute, the attribute_not_exists function will only succeed if no matching item exists.

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.
- **return_values** (*bool*) – Set to `True` if you want to get the item attributes as they appeared before they were updated with the *PutItem* request.
- **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** (*bool*) – Determines whether item collection metrics are returned.

Return type dict

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

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

You can use the `scan_index_forward` parameter to get results in forward or reverse order, by sort key.

Queries that do not return results consume the minimum number of read capacity units for that type of read operation.

If the total number of items meeting the query criteria exceeds the result set size limit of 1 MB, the query stops and results are returned to the user with the `LastEvaluatedKey` element to continue the query in a subsequent operation. Unlike a *Scan* operation, a Query operation never returns both an empty result set and a `LastEvaluatedKey` value. `LastEvaluatedKey` is only provided if the results exceed 1 MB, or if you have used the `limit` parameter.

You can query a table, a local secondary index, or a global secondary index. For a query on a table or on a local secondary index, you can set the `consistent_read` parameter to true and obtain a strongly consistent result. Global secondary indexes support eventually consistent reads only, so do not specify `consistent_read` when querying a global secondary 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** (*str|bytes|int*) – 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*, *consistent_read=False*, *exclusive_start_key=None*, *expression_attribute_names=None*, *expression_attribute_values=None*, *filter_expression=None*, *projection_expression=None*, *index_name=None*, *limit=None*, *return_consumed_capacity=None*, *segment=None*, *total_segments=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.

Parameters

- **table_name** (*str*) – The name of the table containing the requested items; or, if you provide IndexName, the name of the table to which that index belongs.
- **consistent_read** (*bool*) – A Boolean value that determines the read consistency model during the scan:
  - If set to False, then the data returned from *Scan* might not contain the results from other recently completed write operations (*PutItem*, *UpdateItem*, or *DeleteItem*).
  - If set to True, then all of the write operations that completed before the Scan began are guaranteed to be contained in the *Scan* response.

  The default setting is False.

  This parameter is not supported on global secondary indexes. If you scan a global secondary index and set consistent_read to true, you will receive a *ValidationException*.
- **exclusive_start_key** (*str|bytes|int*) – 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 Scan operation, but before the data is returned to you. Items that do not satisfy the expression criteria are not returned.

  Note: 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*) – A string that identifies one or more attributes to retrieve from the specified table or index. 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.

  For more information, see Accessing Item Attributes in the Amazon DynamoDB Developer Guide.
- **index_name** (*str*) – The name of a secondary index to scan. 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. Should be `None` or one of `INDEXES` or `TOTAL`
- **segment** (*int*) – For a parallel *Scan* request, `segment` identifies an individual segment to be scanned by an application worker.

  Segment IDs are zero-based, so the first segment is always `0`. For example, if you want to use four application threads to scan a table or an index, then the first thread specifies a Segment value of `0`, the second thread specifies `1`, and so on.

  The value of `LastEvaluatedKey` returned from a parallel *Scan* request must be used as `ExclusiveStartKey` with the same segment ID in a subsequent *Scan* operation.

  The value for `segment` must be greater than or equal to `0`, and less than the value provided for `total_segments`.

  If you provide `segment`, you must also provide `total_segments`.
- **total_segments** (*int*) – For a parallel *Scan* request, `total_segments` represents the total number of segments into which the *Scan* operation will be divided. The value of `total_segments` corresponds to the number of application workers that will perform

the parallel scan. For example, if you want to use four application threads to scan a table or an index, specify a `total_segments` value of 4.

The value for `total_segments` must be greater than or equal to `1`, and less than or equal to `1000000`. If you specify a `total_segments` value of `1`, the *Scan* operation will be sequential rather than parallel.

If you specify `total_segments`, you must also specify `segments`.

Return type dict

update_item(*table_name*, *key*, *return_values=False*, *condition_expression=None*, *update_expression=None*, *expression_attribute_names=None*, *expression_attribute_values=None*, *return_consumed_capacity=None*, *return_item_collection_metrics=False*)

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*) – 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.
- **return_values** (*bool*) – Set to `True` if you want to get the item attributes as they appeared before they were updated with the *UpdateItem* request.
- **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. Should be `None` or one of `INDEXES` or `TOTAL`
- **return_item_collection_metrics** (*bool*) – Determines whether item collection metrics are returned.

Return type dict

update_table(*name*, *attributes*, *read_capacity_units=1*, *write_capacity_units=1*, *global_secondary_index_updates=None*, *stream_enabled=False*, *stream_view_type=None*)

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 backfilling, 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

- **name** (*str*) – The table name to be updated
- **attributes** (*list*) – A list of attribute definition key/value pairs where the key is the name of the attribute and the value is one of `S`, `N`, or `B` indicating the data type of the attribute. If you are adding a new global secondary index to the table, the attributes must include the key element(s) of the new index.
- **read_capacity_units** (*int*) – The maximum number of strongly consistent reads consumed per second before DynamoDB returns a *ThrottlingException*
- **write_capacity_units** (*int*) – The maximum number of writes consumed per second before DynamoDB returns a *ThrottlingException*
- **global_secondary_index_updates** (*dict*) – An array of one or more global secondary indexes for the table. For each index in the array, you can request one action:
  - `Create`: add a new global secondary index to the table.
  - **Update: modify the provisioned throughput settings of an** existing global secondary index.
  - `Delete`: remove a global secondary index from the table.

  For more information, see Managing Global Secondary Indexes in the Amazon DynamoDB Developer Guide.
- **stream_enabled** (*bool*) – Indicates whether DynamoDB Streams is enabled (`True`) or disabled (`False`) for the table.
- **stream_view_type** (*str*) – When an item in the table is modified, StreamViewType determines what information is written to the stream for this table.

Return type dict

CHAPTER 2

Exceptions

DynamoDB Exceptions

exception tornado_dynamodb.exceptions.**ConditionalCheckFailedException**(**args*, ***kwargs*)

A condition specified in the operation could not be evaluated.

exception tornado_dynamodb.exceptions.**ConfigNotFound**(**args*, ***kwargs*)

The configuration file could not be parsed.

exception tornado_dynamodb.exceptions.**ConfigParserError**(**args*, ***kwargs*)

Error raised when parsing a configuration file with RawConfigParser

exception tornado_dynamodb.exceptions.**DynamoDBException**(**args*, ***kwargs*)

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

Variables **msg** – The error message

exception tornado_dynamodb.exceptions.**InternalFailure**(**args*, ***kwargs*)

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

exception tornado_dynamodb.exceptions.**InvalidAction**(**args*, ***kwargs*)

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

exception tornado_dynamodb.exceptions.**InvalidParameterCombination**(**args*, ***kwargs*)

Parameters that must not be used together were used together.

exception tornado_dynamodb.exceptions.**InvalidParameterValue**(**args*, ***kwargs*)

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

exception tornado_dynamodb.exceptions.**InvalidQueryParameter**(**args*, ***kwargs*)

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

exception tornado_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 tornado_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 tornado_dynamodb.exceptions.**MalformedQueryString**(**args*, ***kwargs*)
The query string contains a syntax error.

exception tornado_dynamodb.exceptions.**MissingParameter**(**args*, ***kwargs*)
A required parameter for the specified action is not supplied.

exception tornado_dynamodb.exceptions.**NoCredentialsError**(**args*, ***kwargs*)
Raised when the credentials could not be located.

exception tornado_dynamodb.exceptions.**NoProfileError**(**args*, ***kwargs*)
Raised when the specified profile could not be located.

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

exception tornado_dynamodb.exceptions.**ProvisionedThroughputExceeded**(**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 tornado_dynamodb.exceptions.**RequestException**(**args*, ***kwargs*)
A generic HTTP request exception has occurred when communicating with DynamoDB.

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

exception tornado_dynamodb.exceptions.**ThrottlingException**(**args*, ***kwargs*)
The request was denied due to request throttling.

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

exception tornado_dynamodb.exceptions.**ValidationException**(**args*, ***kwargs*)
The input fails to satisfy the constraints specified by an AWS service.

CHAPTER 3

Examples

The following example uses invokes the `DescribeTable` endpoint for DynamoDB:

```
import json
import pprint

import tornado_aws
from tornado import gen, ioloop

HEADERS = {'Content-Type': 'application/x-amz-json-1.0',
           'x-amz-target': 'DynamoDB_20120810.DescribeTable'}
PAYLOAD = {'TableName': 'prod-us-east-1-history-events'}

@gen.coroutine
def async_request():
    client = tornado_aws.AsyncAWSClient('dynamodb')
    response = yield client.fetch('POST', '/', headers=HEADERS,
                                  body=json.dumps(PAYLOAD))
    x = json.loads(response.body.decode('utf-8'))
    pprint.pprint(x)
    ioloop.IOLoop.instance().stop()

_ioloop = ioloop.IOLoop.instance()
_ioloop.add_callback(async_request)
_ioloop.start()
```

CHAPTER 4

Issues

Please report any issues to the Github repo at https://github.com/gmr/tornado-dynamodb/issues

CHAPTER 5

Source

tornado-dynamodb source is available on Github at https://github.com/gmr/tornado-dynamodb

CHAPTER 6

Version History

See history

CHAPTER 7

Indices and tables

- genindex
- modindex
- search

Python Module Index

t

Index