# The DynamoDB module

## Getting started

To start using the module, you need to load it. Since it is built into the base QDL distribution, it is quite easy:

```
   ddb := j_load('dynamo');
```

Since there is a default region set on startup, you can test if it loaded right by querying that:

```
   ddb#region()
us-east-2a
```

## Typical example session

```
   dd := j_load('dynamo');

   // Setup method #1, open the connection with all the parameters
       cfg.'access_key_id' := 'YOUR_KEY';
   cfg.'secret_access_key' := 'YOUR_SECRET';
          cfg.'table_name' := 'fermilab_dev';
       cfg.'partition_key' := 'PartitionKey';
   dd#open(cfg.);

// Setup method #2, set a bunch of them using mutators
   dd#region('us-east-2');
   dd#table_name('fermilab_dev');
   dd#partition_key('PartitionKey');
   dd#open('YOUR_KEY', 'YOUR_SECRET');

// either way, it is ready to be used. The main method is get:
   out. := dd#get('identifier#eppn#dwd@fnal.gov');
   out.; // print it
   {  PartitionKey : identifier#eppn#dwd@fnal.gov,
        cm_members : [/dune,/fermilab],
    capabilitysets : [wlcg.capabilityset:/duneana,wlcg.capabilityset:/fermilab]
    }
   // pretty print a specific capability set, using get again:
   // dd#get retrieve a set of capabilities as a stem, then we access them

   print( dd#get(out.'capabilitysets'.0).'capabilities');
0 : compute.cancel
1 : compute.create
2 : compute.modify
3 : compute.read
4 : storage.create:/dune/resilient/jobsub_stage
5 : storage.create:/dune/scratch/users/${uid}
6 : storage.create:/fermigrid/jobsub/jobs
7 : storage.read:/dune
8 : storage.read:/dune/resilient/jobsub_stage
```

# Operations

## Close

### Description

Close the current session. You must class **open** again before doing any operations.

### Usage

```
close()
```

### Arguments

none

### Output

Logical true if it worked, or an error is raised.

## Get

### Description

Get the entry for a given key

### Usage

```
get(key)
```

### Arguments

key – a string that is the key for the entry

### Output

A stem with the entries. Note that the entries are converted to QDL types, so you will get back, booleans, integers, decimals, lists and stems. If there is no such entry, the result is empty.

# Open

## Description

Open a connection to a dynamo database.

## Usage

```
open(cfg.)
open(access_key_id, secret_access_key)
```

## Arguments

If you want to initialize the connection in one go, then use a cfg. Stem. The key/value pairs are

| Name | Req? | Description |
|---|---|---|
| access_key_id | Y | The access key. Part of your AWS credentials |
| secret_access_key | Y | The secret access key. Part of your AWS credentials |
| region | N | The AWS region |
| table_name | N | The table name for all queries. Can be set |
| partition_key | N | The partition key for all queries. Can be set |

otherwise, if you have set the region, table name and partition key separately (there are methods for those), then you can just open the connection using the access key id and secret access key.

## Output

A boolean true if it succeeded otherwise an error is raised.

## Examples

See the example session above for both ways to open a connection.

# partition_key

Set or query the partition key for the connection. This must be set before issuing the open() call.

## Usage

```
partition_key({new_key})
```

## Arguments

none – simply query the current key, if any.

new_key – the new key to use. The old key is returned

## Region

Set or query the region for the connection. Note this has a default value of **`us_east_2`**.

### Usage

```
region({new_region})
```

### Arguments

none – query current region. A default is usually set.

new_region – the new region. The old region is returned.

## Regionsm

Query the supported regions for this module.

### Usage

```
regions()
```

### Arguments

none

### Output

All supported regions, including possibly other information.

## table_name

Set or query the tablename used for the connection

### Usage

```
table_name({new_table_name})
```

### Arguments

none – query current table name

new_table_name – the new table name to use. The old one is returned.