**An Introduction to boto’s SimpleDB interface**

This tutorial focuses on the boto interface to AWS’ SimpleDB. This tutorial assumes that you have boto already downloaded and installed.

**Note**

If you’re starting a new application, you might want to consider using DynamoDB2 instead, as it has a more comprehensive feature set & has guaranteed performance throughput levels.

Only 6 AWS managed policies mention sdb explicitly:

- AmazonElasticMapReduceReadOnlyAccess

- AmazonElasticMapReduceRole

- AmazonElasticMapReduceforEC2Role

- ReadOnlyAccess

- AmazonElasticMapReduceFullAccess

- SecurityAudit

**Creating a Connection**

The first step in accessing SimpleDB is to create a connection to the service. To do so, the most straight forward way is the following:

**>>> import** **boto.sdb**

**>>>** conn = boto.sdb.connect\_to\_region(

**...**  'us-west-2',

**...**  aws\_access\_key\_id='<YOUR\_AWS\_KEY\_ID>',

**...**  aws\_secret\_access\_key='<YOUR\_AWS\_SECRET\_KEY>')

**>>>** conn

SDBConnection:sdb.amazonaws.com

>>>

Bear in mind that if you have your credentials in boto config in your home directory, the two keyword arguments in the call above are not needed. Also important to note is that just as any other AWS service, SimpleDB is region-specific and as such you might want to specify which region to connect to, by default, it’ll connect to the US-EAST-1 region.

**Creating Domains**

Arguably, once you have your connection established, you’ll want to create one or more dmains. Creating new domains is a fairly straight forward operation. To do so, you can proceed as follows:

**>>>** conn.create\_domain('test-domain')

Domain:test-domain

>>>

**>>>** conn.create\_domain('test-domain-2')

Domain:test-domain

>>>

Please note that SimpleDB, unlike its newest sibling DynamoDB, is truly and completely schema-less. Thus, there’s no need specify domain keys or ranges.

**Listing All Domains**

Unlike DynamoDB or other database systems, SimpleDB uses the concept of ‘domains’ instead of tables. So, to list all your domains for your account in a region, you can simply do as follows:

**>>>** domains = conn.get\_all\_domains()

**>>>** domains

[Domain:test-domain, Domain:test-domain-2]

>>>

The get\_all\_domains() method returns a **boto.resultset.ResultSet** containing all **boto.sdb.domain.Domain** objects associated with this connection’s Access Key ID for that region.

**Retrieving a Domain (by name)**

If you wish to retrieve a specific domain whose name is known, you can do so as follows:

**>>>** dom = conn.get\_domain('test-domain')

**>>>** dom

Domain:test-domain

>>>

The get\_domain call has an optional validate parameter, which defaults to True. This will make sure to raise an exception if the domain you are looking for doesn’t exist. If you set it to false, it will return a **Domain** object blindly regardless of its existence.

**Getting Domain Metadata**

There are times when you might want to know your domains’ machine usage, aprox. item count and other such data. To this end, boto offers a simple and convenient way to do so as shown below:

**>>>** domain\_meta = conn.domain\_metadata(dom)

**>>>** domain\_meta

<boto.sdb.domain.DomainMetaData instance at 0x23cd440>

**>>>** dir(domain\_meta)

['BoxUsage', 'DomainMetadataResponse', 'DomainMetadataResult', 'RequestId', 'ResponseMetadata',

'\_\_doc\_\_', '\_\_init\_\_', '\_\_module\_\_', 'attr\_name\_count', 'attr\_names\_size', 'attr\_value\_count', 'attr\_values\_size',

'domain', 'endElement', 'item\_count', 'item\_names\_size', 'startElement', 'timestamp']

**>>>** domain\_meta.item\_count

0

>>>

Please bear in mind that while in the example above we used a previously retrieved domain object as the parameter, you can retrieve the domain metadata via its name (string).

**Adding Items (and attributes)**

Once you have your domain setup, presumably, you’ll want to start adding items to it. In its most straight forward form, you need to provide a name for the item – think of it as a record id – and a collection of the attributes you want to store in the item (often a Dictionary-like object). So, adding an item to a domain looks as follows:

**>>>** item\_name = 'ABC\_123'

**>>>** item\_attrs = {'Artist': 'The Jackson 5', 'Genera':'Pop'}

**>>>** dom.put\_attributes(item\_name, item\_attrs)

True

>>>

Now let’s check if it worked:

**>>>** domain\_meta = conn.domain\_metadata(dom)

**>>>** domain\_meta.item\_count

1

>>>

**Batch Adding Items (and attributes)**

You can also add a number of items at the same time in a similar fashion. All you have to provide to the batch\_put\_attributes() method is a Dictionary-like object with your items and their respective attributes, as follows:

**>>>** items = {'item1':{'attr1':'val1'},'item2':{'attr2':'val2'}}

**>>>** dom.batch\_put\_attributes(items)

True

>>>

Now, let’s check the item count once again:

**>>>** domain\_meta = conn.domain\_metadata(dom)

**>>>** domain\_meta.item\_count

3

>>>

A few words of warning: both batch\_put\_attributes() and put\_item(), by default, will overwrite the values of the attributes if both the item and attribute already exist. If the item exists, but not the attributes, it will append the new attributes to the attribute list of that item. If you do not wish these methods to behave in that manner, simply supply them with a ‘replace=False’ parameter.

**Retrieving Items**

To retrieve an item along with its attributes is a fairly straight forward operation and can be accomplished as follows:

**>>>** dom.get\_item('item1')

{u'attr1': u'val1'}

>>>

Since SimpleDB works in an “eventual consistency” manner, we can also request a forced consistent read (though this will invariably adversely affect read performance). The way to accomplish that is as shown below:

**>>>** dom.get\_item('item1', consistent\_read=**True**)

{u'attr1': u'val1'}

>>>

**Retrieving One or More Items**

Another way to retrieve items is through boto’s select() method. This method, at the bare minimum, requires a standard SQL select query string and you would do something along the lines of:

**>>>** query = 'select \* from `test-domain` where attr1="val1"'

**>>>** rs = dom.select(query)

**>>> for** j **in** rs:

**...**  print 'o hai'

**...**

o hai

>>>

This method returns a ResultSet collection you can iterate over.

**Updating Item Attributes**

The easiest way to modify an item’s attributes is by manipulating the item’s attributes and then saving those changes. For example:

**>>>** item = dom.get\_item('item1')

**>>>** item['attr1'] = 'val\_changed'

**>>>** item.save()

**Deleting Items (and its attributes)**

Deleting an item is a very simple operation. All you are required to provide is either the name of the item or an item object to the delete\_item() method, boto will take care of the rest:

>>>dom.delete\_item(item)

>>>True

**Deleting Domains**

To delete a domain and all items under it (i.e. be very careful), you can do it as follows:

**>>>** conn.delete\_domain('test-domain')

True

>>>