boto Documentation

Release HEAD

Mitch Garnaat

May 18, 2017

Contents

An integrated interface to current and future infrastructural services offered by Amazon Web Services.

CHAPTER 1

Currently Supported Services

- **Compute**
  - *Elastic Compute Cloud (EC2)* – (*API Reference*)
  - *Elastic MapReduce (EMR)* – (*API Reference*)
  - *Auto Scaling* – (*API Reference*)
- **Content Delivery**
  - *CloudFront* – (*API Reference*)
- **Database**
  - *SimpleDB* – (*API Reference*)
  - *DynamoDB* – (*API Reference*)
  - Relational Data Services (RDS) – (*API Reference*)
- **Deployment and Management**
  - CloudFormation – (*API Reference*)
- **Identity & Access**
  - Identity and Access Management (IAM) – (*API Reference*)
- **Messaging**
  - *Simple Queue Service (SQS)* – (*API Reference*)
  - Simple Notification Service (SNS) – (*API Reference*)
  - *Simple Email Service (SES)* – (*API Reference*)
- **Monitoring**
  - *CloudWatch* – (*API Reference*)
- **Networking**
  - Route 53 – (*API Reference*)

   - *Virtual Private Cloud (VPC)* – (*API Reference*)
   - *Elastic Load Balancing (ELB)* – (*API Reference*)
- **Payments & Billing**
   - Flexible Payments Service (FPS) – (*API Reference*)
- **Storage**
   - *Simple Storage Service (S3)* – (*API Reference*)
- **Workforce**
   - Mechanical Turk – (*API Reference*)

CHAPTER 2

Additional Resources

- *Boto Config Tutorial*
- Boto Source Repository
- Boto Issue Tracker
- Boto Twitter
- Follow Mitch on Twitter
- Join our IRC channel (#boto on FreeNode).

An Introduction to boto's EC2 interface

This tutorial focuses on the boto interface to the Elastic Compute Cloud from Amazon Web Services. This tutorial assumes that you have already downloaded and installed boto.

Creating a Connection

The first step in accessing EC2 is to create a connection to the service. There are two ways to do this in boto. The first is:

```
>>> from boto.ec2.connection import EC2Connection
>>> conn = EC2Connection('<AWS_ACCESS_KEY_ID>', '<AWS_SECRET_ACCESS_KEY>')
```

At this point the variable conn will point to an EC2Connection object. In this example, the AWS access key and AWS secret key are passed in to the method explicitely. Alternatively, you can set the boto config environment variables and then call the constructor without any arguments, like this:

```
>>> conn = EC2Connection()
```

There is also a shortcut function in the boto package, called connect_ec2 that may provide a slightly easier means of creating a connection:

```
>>> import boto
>>> conn = boto.connect_ec2()
```

In either case, conn will point to an EC2Connection object which we will use throughout the remainder of this tutorial.

Launching Instances

Possibly, the most important and common task you'll use EC2 for is to launch, stop and terminate instances. In its most primitive form, you can launch an instance as follows:

```
>>> conn.run_instances('<ami-image-id>')
```

This will launch an instance in the specified region with the default parameters. You will not be able to SSH into this machine, as it doesn't have a security group set. See *EC2 Security Groups* for details on creating one.

Now, let's say that you already have a key pair, want a specific type of instance, and you have your *security group* all setup. In this case we can use the keyword arguments to accomplish that:

```
>>> conn.run_instances(
        '<ami-image-id>',
        key_name='myKey',
        instance_type='c1.xlarge',
        security_groups=['your-security-group-here'])
```

The main caveat with the above call is that it is possible to request an instance type that is not compatible with the provided AMI (for example, the instance was created for a 64-bit instance and you choose a m1.small instance_type). For more details on the plethora of possible keyword parameters, be sure to check out boto's *EC2 API reference*.

Stopping Instances

Once you have your instances up and running, you might wish to shut them down if they're not in use. Please note that this will only de-allocate virtual hardware resources (as well as instance store drives), but won't destroy your EBS volumes – this means you'll pay nominal provisioned EBS storage fees even if your instance is stopped. To do this, you can do so as follows:

```
>>> conn.stop_instances(instance_ids=['instance-id-1','instance-id-2', ...])
```

This will request a 'graceful' stop of each of the specified instances. If you wish to request the equivalent of unplugging your instance(s), simply add `force=True` keyword argument to the call above. Please note that stop instance is not allowed with Spot instances.

Terminating Instances

Once you are completely done with your instance and wish to surrender both virtual hardware, root EBS volume and all other underlying components you can request instance termination. To do so you can use the call bellow:

```
>>> conn.terminate_instances(instance_ids=['instance-id-1','instance-id-2', ...])
```

Please use with care since once you request termination for an instance there is no turning back.

EC2 Security Groups

Amazon defines a security group as:

“A security group is a named collection of access rules. These access rules specify which ingress, i.e. incoming, network traffic should be delivered to your instance.”

To get a listing of all currently defined security groups:

```
>>> rs = conn.get_all_security_groups()
>>> print rs
[SecurityGroup:appserver, SecurityGroup:default, SecurityGroup:vnc,␣
↪SecurityGroup:webserver]
```

Each security group can have an arbitrary number of rules which represent different network ports which are being enabled. To find the rules for a particular security group, use the rules attribute:

```
>>> sg = rs[1]
>>> sg.name
u'default'
>>> sg.rules
[IPPermissions:tcp(0-65535),
 IPPermissions:udp(0-65535),
 IPPermissions:icmp(-1--1),
 IPPermissions:tcp(22-22),
 IPPermissions:tcp(80-80)]
```

In addition to listing the available security groups you can also create a new security group. I’ll follow through the “Three Tier Web Service” example included in the EC2 Developer’s Guide for an example of how to create security groups and add rules to them.

First, let’s create a group for our Apache web servers that allows HTTP access to the world:

```
>>> web = conn.create_security_group('apache', 'Our Apache Group')
>>> web
SecurityGroup:apache
>>> web.authorize('tcp', 80, 80, '0.0.0.0/0')
True
```

The first argument is the ip protocol which can be one of; tcp, udp or icmp. The second argument is the FromPort or the beginning port in the range, the third argument is the ToPort or the ending port in the range and the last argument is the CIDR IP range to authorize access to.

Next we create another group for the app servers:

```
>>> app = conn.create_security_group('appserver', 'The application tier')
```

We then want to grant access between the web server group and the app server group. So, rather than specifying an IP address as we did in the last example, this time we will specify another SecurityGroup object.:

```
>>> app.authorize(src_group=web)
True
```

Now, to verify that the web group now has access to the app servers, we want to temporarily allow SSH access to the web servers from our computer. Let’s say that our IP address is 192.168.1.130 as it is in the EC2 Developer Guide. To enable that access:

```
>>> web.authorize(ip_protocol='tcp', from_port=22, to_port=22, cidr_ip='192.168.1.130/
↪32')
True
```

Now that this access is authorized, we could ssh into an instance running in the web group and then try to telnet to specific ports on servers in the appserver group, as shown in the EC2 Developer's Guide. When this testing is complete, we would want to revoke SSH access to the web server group, like this:

```
>>> web.rules
[IPPermissions:tcp(80-80),
 IPPermissions:tcp(22-22)]
>>> web.revoke('tcp', 22, 22, cidr_ip='192.168.1.130/32')
True
>>> web.rules
[IPPermissions:tcp(80-80)]
```

EC2

boto.ec2

This module provides an interface to the Elastic Compute Cloud (EC2) service from AWS.

`boto.ec2.`**`connect_to_region`**`(`*region_name*, ***kw_params*`)`

Given a valid region name, return a *`boto.ec2.connection.EC2Connection`*. Any additional parameters after the region_name are passed on to the connect method of the region object.

Type str

Parameters **`region_name`** – The name of the region to connect to.

Return type *`boto.ec2.connection.EC2Connection`* or `None`

Returns A connection to the given region, or None if an invalid region name is given

`boto.ec2.`**`get_region`**`(`*region_name*, ***kw_params*`)`

Find and return a `boto.ec2.regioninfo.RegionInfo` object given a region name.

Type str

Param The name of the region.

Return type `boto.ec2.regioninfo.RegionInfo`

Returns The RegionInfo object for the given region or None if an invalid region name is provided.

`boto.ec2.`**`regions`**`(`***kw_params*`)`

Get all available regions for the EC2 service. You may pass any of the arguments accepted by the EC2Connection object's constructor as keyword arguments and they will be passed along to the EC2Connection object.

Return type *list*

Returns A list of `boto.ec2.regioninfo.RegionInfo`

boto.ec2.address

Represents an EC2 Elastic IP Address

class `boto.ec2.address.`**`Address`**(*connection=None*, *public_ip=None*, *instance_id=None*)

`associate`(*instance_id*)

`delete`()

`disassociate`()

`endElement`(*name*, *value*, *connection*)

`release`()

boto.ec2.autoscale

See the *Auto Scaling Reference*.

boto.ec2.buyreservation

class `boto.ec2.buyreservation.`**`BuyReservation`**

`get`(*params*)

`get_instance_type`(*params*)

`get_quantity`(*params*)

`get_region`(*params*)

`get_zone`(*params*)

boto.ec2.cloudwatch

See the *CloudWatch Reference*.

boto.ec2.connection

Represents a connection to the EC2 service.

class `boto.ec2.connection.`**`EC2Connection`**(*aws_access_key_id=None*, *aws_secret_access_key=None*, *is_secure=True*, *host=None*, *port=None*, *proxy=None*, *proxy_port=None*, *proxy_user=None*, *proxy_pass=None*, *debug=0*, *https_connection_factory=None*, *region=None*, *path='/'*, *api_version=None*, *security_token=None*)

Init method to create a new connection to EC2.

`APIVersion` = **'2011-12-15'**

`DefaultRegionEndpoint` = **'ec2.us-east-1.amazonaws.com'**

`DefaultRegionName` = **'us-east-1'**

`ResponseError`

alias of `EC2ResponseError`

allocate_address(*domain=None*)

Allocate a new Elastic IP address and associate it with your account.

Return type *boto.ec2.address.Address*

Returns The newly allocated Address

associate_address(*instance_id*, *public_ip=None*, *allocation_id=None*)

Associate an Elastic IP address with a currently running instance. This requires one of `public_ip` or `allocation_id` depending on if you're associating a VPC address or a plain EC2 address.

Parameters

- **instance_id** (*string*) – The ID of the instance
- **public_ip** (*string*) – The public IP address for EC2 based allocations.
- **allocation_id** (*string*) – The allocation ID for a VPC-based elastic IP.

Return type bool

Returns True if successful

attach_network_interface(*network_interface_id*, *instance_id*, *device_index*)

Attaches a network interface to an instance.

Parameters

- **network_interface_id** (*str*) – The ID of the network interface to attach.
- **instance_id** (*str*) – The ID of the instance that will be attached to the network interface.
- **device_index** (*int*) – The index of the device for the network interface attachment on the instance.

attach_volume(*volume_id*, *instance_id*, *device*)

Attach an EBS volume to an EC2 instance.

Parameters

- **volume_id** (*str*) – The ID of the EBS volume to be attached.
- **instance_id** (*str*) – The ID of the EC2 instance to which it will be attached.
- **device** (*str*) – The device on the instance through which the volume will be exposted (e.g. /dev/sdh)

Return type bool

Returns True if successful

authorize_security_group(*group_name=None*, *src_security_group_name=None*, *src_security_group_owner_id=None*, *ip_protocol=None*, *from_port=None*, *to_port=None*, *cidr_ip=None*, *group_id=None*, *src_security_group_group_id=None*)

Add a new rule to an existing security group. You need to pass in either src_security_group_name and src_security_group_owner_id OR ip_protocol, from_port, to_port, and cidr_ip. In other words, either you are authorizing another group or you are authorizing some ip-based rule.

Parameters

- **group_name** (*string*) – The name of the security group you are adding the rule to.
- **src_security_group_name** (*string*) – The name of the security group you are granting access to.

- **src_security_group_owner_id** (*string*) – The ID of the owner of the security group you are granting access to.
- **ip_protocol** (*string*) – Either tcp | udp | icmp
- **from_port** (*int*) – The beginning port number you are enabling
- **to_port** (*int*) – The ending port number you are enabling
- **cidr_ip** (*string*) – The CIDR block you are providing access to. See http://goo.gl/Yj5QC
- **group_id** (*string*) – ID of the EC2 or VPC security group to modify. This is required for VPC security groups and can be used instead of group_name for EC2 security groups.
- **group_id** – ID of the EC2 or VPC source security group. This is required for VPC security groups and can be used instead of group_name for EC2 security groups.

Return type bool

Returns True if successful.

authorize_security_group_deprecated (*group_name*, *src_security_group_name=None*, *src_security_group_owner_id=None*, *ip_protocol=None*, *from_port=None*, *to_port=None*, *cidr_ip=None*)

NOTE: This method uses the old-style request parameters that did not allow a port to be specified when authorizing a group.

Parameters

- **group_name** (*string*) – The name of the security group you are adding the rule to.
- **src_security_group_name** (*string*) – The name of the security group you are granting access to.
- **src_security_group_owner_id** (*string*) – The ID of the owner of the security group you are granting access to.
- **ip_protocol** (*string*) – Either tcp | udp | icmp
- **from_port** (*int*) – The beginning port number you are enabling
- **to_port** (*string*) – The ending port number you are enabling
- **to_port** – The CIDR block you are providing access to. See http://goo.gl/Yj5QC

Return type bool

Returns True if successful.

authorize_security_group_egress (*group_id*, *ip_protocol*, *from_port=None*, *to_port=None*, *src_group_id=None*, *cidr_ip=None*)

The action adds one or more egress rules to a VPC security group. Specifically, this action permits instances in a security group to send traffic to one or more destination CIDR IP address ranges, or to one or more destination security groups in the same VPC.

build_filter_params (*params*, *filters*)

build_tag_param_list (*params*, *tags*)

bundle_instance (*instance_id*, *s3_bucket*, *s3_prefix*, *s3_upload_policy*)

Bundle Windows instance.

Parameters

- **instance_id** (*string*) – The instance id
- **s3_bucket** (*string*) – The bucket in which the AMI should be stored.
- **s3_prefix** (*string*) – The beginning of the file name for the AMI.
- **s3_upload_policy** (*string*) – Base64 encoded policy that specifies condition and permissions for Amazon EC2 to upload the user's image into Amazon S3.

cancel_bundle_task (*bundle_id*)

Cancel a previously submitted bundle task

Parameters **bundle_id** (*string*) – The identifier of the bundle task to cancel.

cancel_spot_instance_requests (*request_ids*)

Cancel the specified Spot Instance Requests.

Parameters **request_ids** (list) – A list of strings of the Request IDs to terminate

Return type *list*

Returns A list of the instances terminated

confirm_product_instance (*product_code*, *instance_id*)

create_image (*instance_id*, *name*, *description=None*, *no_reboot=False*)

Will create an AMI from the instance in the running or stopped state.

Parameters

- **instance_id** (*string*) – the ID of the instance to image.
- **name** (*string*) – The name of the new image
- **description** (*string*) – An optional human-readable string describing the contents and purpose of the AMI.
- **no_reboot** (*bool*) – An optional flag indicating that the bundling process should not attempt to shutdown the instance before bundling. If this flag is True, the responsibility of maintaining file system integrity is left to the owner of the instance.

Return type string

Returns The new image id

create_key_pair (*key_name*)

Create a new key pair for your account. This will create the key pair within the region you are currently connected to.

Parameters **key_name** (*string*) – The name of the new keypair

Return type *boto.ec2.keypair.KeyPair*

Returns The newly created *boto.ec2.keypair.KeyPair*. The material attribute of the new KeyPair object will contain the the unencrypted PEM encoded RSA private key.

create_network_interface (*subnet_id*, *private_ip_address=None*, *description=None*, *groups=None*)

Creates a network interface in the specified subnet.

Parameters

- **subnet_id** (*str*) – The ID of the subnet to associate with the network interface.
- **private_ip_address** (*str*) – The private IP address of the network interface. If not supplied, one will be chosen for you.
- **description** (*str*) – The description of the network interface.

- **groups** (*list*) – Lists the groups for use by the network interface. This can be either a list of group ID's or a list of *boto.ec2.securitygroup.SecurityGroup* objects.

Return type boto.ec2.networkinterface.NetworkInterface

Returns The newly created network interface.

create_placement_group (*name*, *strategy='cluster'*)
Create a new placement group for your account. This will create the placement group within the region you are currently connected to.

Parameters

- **name** (*string*) – The name of the new placement group
- **strategy** (*string*) – The placement strategy of the new placement group. Currently, the only acceptable value is "cluster".

Return type bool

Returns True if successful

create_security_group (*name*, *description*, *vpc_id=None*)
Create a new security group for your account. This will create the security group within the region you are currently connected to.

Parameters

- **name** (*string*) – The name of the new security group
- **description** (*string*) – The description of the new security group
- **vpc_id** (*string*) – The ID of the VPC to create the security group in, if any.

Return type *boto.ec2.securitygroup.SecurityGroup*

Returns The newly created *boto.ec2.keypair.KeyPair*.

create_snapshot (*volume_id*, *description=None*)
Create a snapshot of an existing EBS Volume.

Parameters

- **volume_id** (*str*) – The ID of the volume to be snapshot'ed
- **description** (*str*) – A description of the snapshot. Limited to 255 characters.

Return type bool

Returns True if successful

create_spot_datafeed_subscription (*bucket*, *prefix*)
Create a spot instance datafeed subscription for this account.

Parameters

- **bucket** (*str or unicode*) – The name of the bucket where spot instance data will be written. The account issuing this request must have FULL_CONTROL access to the bucket specified in the request.
- **prefix** (*str or unicode*) – An optional prefix that will be pre-pended to all data files written to the bucket.

Return type boto.ec2.spotdatafeedsubscription.SpotDatafeedSubscription

Returns The datafeed subscription object or None

create_tags (*resource_ids*, *tags*)
Create new metadata tags for the specified resource ids.

Parameters

- **resource_ids** (list) – List of strings
- **tags** (dict) – A dictionary containing the name/value pairs. If you want to create only a tag name, the value for that tag should be the empty string (e.g. '').

create_volume (*size*, *zone*, *snapshot=None*)
Create a new EBS Volume.

Parameters

- **size** (*int*) – The size of the new volume, in GiB
- **zone** (string or *boto.ec2.zone.Zone*) – The availability zone in which the Volume will be created.
- **snapshot** (string or *boto.ec2.snapshot.Snapshot*) – The snapshot from which the new Volume will be created.

delete_key_pair (*key_name*)
Delete a key pair from your account.

Parameters **key_name** (*string*) – The name of the keypair to delete

delete_network_interface (*network_interface_id*)
Delete the specified network interface.

Parameters **network_interface_id** (*str*) – The ID of the network interface to delete.

delete_placement_group (*name*)
Delete a placement group from your account.

Parameters **key_name** (*string*) – The name of the keypair to delete

delete_security_group (*name=None*, *group_id=None*)
Delete a security group from your account.

Parameters

- **name** (*string*) – The name of the security group to delete.
- **group_id** (*string*) – The ID of the security group to delete within a VPC.

Return type bool

Returns True if successful.

delete_snapshot (*snapshot_id*)

delete_spot_datafeed_subscription ()
Delete the current spot instance data feed subscription associated with this account

Return type bool

Returns True if successful

delete_tags (*resource_ids*, *tags*)
Delete metadata tags for the specified resource ids.

Parameters

- **resource_ids** (list) – List of strings

- **tags** (dict or list) – Either a dictionary containing name/value pairs or a list containing just tag names. If you pass in a dictionary, the values must match the actual tag values or the tag will not be deleted. If you pass in a value of None for the tag value, all tags with that name will be deleted.

delete_volume (*volume_id*)
Delete an EBS volume.

Parameters **volume_id** (*str*) – The ID of the volume to be delete.

Return type bool

Returns True if successful

deregister_image (*image_id*, *delete_snapshot=False*)
Unregister an AMI.

Parameters

- **image_id** (*string*) – the ID of the Image to unregister
- **delete_snapshot** (*bool*) – Set to True if we should delete the snapshot associated with an EBS volume mounted at /dev/sda1

Return type bool

Returns True if successful

detach_network_interface (*network_interface_id*, *force=False*)
Detaches a network interface from an instance.

Parameters

- **network_interface_id** (*str*) – The ID of the network interface to detach.
- **force** (*bool*) – Set to true to force a detachment.

detach_volume (*volume_id*, *instance_id=None*, *device=None*, *force=False*)
Detach an EBS volume from an EC2 instance.

Parameters

- **volume_id** (*str*) – The ID of the EBS volume to be attached.
- **instance_id** (*str*) – The ID of the EC2 instance from which it will be detached.
- **device** (*str*) – The device on the instance through which the volume is exposted (e.g. /dev/sdh)
- **force** (*bool*) – Forces detachment if the previous detachment attempt did not occur cleanly. This option can lead to data loss or a corrupted file system. Use this option only as a last resort to detach a volume from a failed instance. The instance will not have an opportunity to flush file system caches nor file system meta data. If you use this option, you must perform file system check and repair procedures.

Return type bool

Returns True if successful

disassociate_address (*public_ip=None*, *association_id=None*)
Disassociate an Elastic IP address from a currently running instance.

Parameters

- **public_ip** (*string*) – The public IP address for EC2 elastic IPs.
- **association_id** (*string*) – The association ID for a VPC based elastic ip.

Return type bool

Returns True if successful

get_all_addresses(*addresses=None*, *filters=None*, *allocation_ids=None*)
Get all EIP's associated with the current credentials.

Parameters

- **addresses** (list) – Optional list of addresses. If this list is present, only the Addresses associated with these addresses will be returned.
- **filters** (dict) – Optional filters that can be used to limit the results returned. Filters are provided in the form of a dictionary consisting of filter names as the key and filter values as the value. The set of allowable filter names/values is dependent on the request being performed. Check the EC2 API guide for details.
- **allocation_ids** (list) – Optional list of allocation IDs. If this list is present, only the Addresses associated with the given allocation IDs will be returned.

Return type list of *boto.ec2.address.Address*

Returns The requested Address objects

get_all_bundle_tasks(*bundle_ids=None*, *filters=None*)
Retrieve current bundling tasks. If no bundle id is specified, all tasks are retrieved.

Parameters

- **bundle_ids** (list) – A list of strings containing identifiers for previously created bundling tasks.
- **filters** (dict) – Optional filters that can be used to limit the results returned. Filters are provided in the form of a dictionary consisting of filter names as the key and filter values as the value. The set of allowable filter names/values is dependent on the request being performed. Check the EC2 API guide for details.

get_all_images(*image_ids=None*, *owners=None*, *executable_by=None*, *filters=None*)
Retrieve all the EC2 images available on your account.

Parameters

- **image_ids** (list) – A list of strings with the image IDs wanted
- **owners** (list) – A list of owner IDs
- **executable_by** (list) – Returns AMIs for which the specified user ID has explicit launch permissions
- **filters** (dict) – Optional filters that can be used to limit the results returned. Filters are provided in the form of a dictionary consisting of filter names as the key and filter values as the value. The set of allowable filter names/values is dependent on the request being performed. Check the EC2 API guide for details.

Return type *list*

Returns A list of *boto.ec2.image.Image*

get_all_instance_status(*instance_ids=None*, *max_results=None*, *next_token=None*, *filters=None*)
Retrieve all the instances in your account scheduled for maintenance.

Parameters

- **instance_ids** (list) – A list of strings of instance IDs

- **`max_results`** (*`int`*) – The maximum number of paginated instance items per response.
- **`next_token`** (*`str`*) – A string specifying the next paginated set of results to return.
- **`filters`** (`dict`) – Optional filters that can be used to limit the results returned. Filters are provided in the form of a dictionary consisting of filter names as the key and filter values as the value. The set of allowable filter names/values is dependent on the request being performed. Check the EC2 API guide for details.

Return type *list*

Returns A list of instances that have maintenance scheduled.

`get_all_instances`(*instance_ids=None*, *filters=None*)
Retrieve all the instances associated with your account.

Parameters

- **`instance_ids`** (`list`) – A list of strings of instance IDs
- **`filters`** (`dict`) – Optional filters that can be used to limit the results returned. Filters are provided in the form of a dictionary consisting of filter names as the key and filter values as the value. The set of allowable filter names/values is dependent on the request being performed. Check the EC2 API guide for details.

Return type *list*

Returns A list of *`boto.ec2.instance.Reservation`*

`get_all_kernels`(*kernel_ids=None*, *owners=None*)
Retrieve all the EC2 kernels available on your account. Constructs a filter to allow the processing to happen server side.

Parameters

- **`kernel_ids`** (`list`) – A list of strings with the image IDs wanted
- **`owners`** (`list`) – A list of owner IDs

Return type *list*

Returns A list of *`boto.ec2.image.Image`*

`get_all_key_pairs`(*keynames=None*, *filters=None*)
Get all key pairs associated with your account.

Parameters

- **`keynames`** (`list`) – A list of the names of keypairs to retrieve. If not provided, all key pairs will be returned.
- **`filters`** (`dict`) – Optional filters that can be used to limit the results returned. Filters are provided in the form of a dictionary consisting of filter names as the key and filter values as the value. The set of allowable filter names/values is dependent on the request being performed. Check the EC2 API guide for details.

Return type *list*

Returns A list of *`boto.ec2.keypair.KeyPair`*

`get_all_network_interfaces`(*filters=None*)
Retrieve all of the Elastic Network Interfaces (ENI's) associated with your account.

Parameters **`filters`** (`dict`) – Optional filters that can be used to limit the results returned. Filters are provided in the form of a dictionary consisting of filter names as the key and filter values as the value. The set of allowable filter names/values is dependent on the request being performed. Check the EC2 API guide for details.

Return type *list*

Returns A list of `boto.ec2.networkinterface.NetworkInterface`

`get_all_placement_groups`(*groupnames=None*, *filters=None*)
Get all placement groups associated with your account in a region.

Parameters

- **`groupnames`** (`list`) – A list of the names of placement groups to retrieve. If not provided, all placement groups will be returned.
- **`filters`** (`dict`) – Optional filters that can be used to limit the results returned. Filters are provided in the form of a dictionary consisting of filter names as the key and filter values as the value. The set of allowable filter names/values is dependent on the request being performed. Check the EC2 API guide for details.

Return type *list*

Returns A list of `boto.ec2.placementgroup.PlacementGroup`

`get_all_ramdisks`(*ramdisk_ids=None*, *owners=None*)
Retrieve all the EC2 ramdisks available on your account. Constructs a filter to allow the processing to happen server side.

Parameters

- **`ramdisk_ids`** (`list`) – A list of strings with the image IDs wanted
- **`owners`** (`list`) – A list of owner IDs

Return type *list*

Returns A list of *`boto.ec2.image.Image`*

`get_all_regions`(*region_names=None*, *filters=None*)
Get all available regions for the EC2 service.

Parameters

- **`region_names`** (*`list of str`*) – Names of regions to limit output
- **`filters`** (`dict`) – Optional filters that can be used to limit the results returned. Filters are provided in the form of a dictionary consisting of filter names as the key and filter values as the value. The set of allowable filter names/values is dependent on the request being performed. Check the EC2 API guide for details.

Return type *list*

Returns A list of `boto.ec2.regioninfo.RegionInfo`

`get_all_reserved_instances`(*reserved_instances_id=None*, *filters=None*)
Describes Reserved Instance offerings that are available for purchase.

Parameters

- **`reserved_instance_ids`** (`list`) – A list of the reserved instance ids that will be returned. If not provided, all reserved instances will be returned.
- **`filters`** (`dict`) – Optional filters that can be used to limit the results returned. Filters are provided in the form of a dictionary consisting of filter names as the key and filter values as the value. The set of allowable filter names/values is dependent on the request being performed. Check the EC2 API guide for details.

Return type *list*

Returns A list of `boto.ec2.reservedinstance.ReservedInstance`

get_all_reserved_instances_offerings(*reserved_instances_id=None*, *instance_type=None*, *availability_zone=None*, *product_description=None*, *filters=None*)

Describes Reserved Instance offerings that are available for purchase.

Parameters

- **reserved_instances_id** (*str*) – Displays Reserved Instances with the specified offering IDs.
- **instance_type** (*str*) – Displays Reserved Instances of the specified instance type.
- **availability_zone** (*str*) – Displays Reserved Instances within the specified Availability Zone.
- **product_description** (*str*) – Displays Reserved Instances with the specified product description.
- **filters** (`dict`) – Optional filters that can be used to limit the results returned. Filters are provided in the form of a dictionary consisting of filter names as the key and filter values as the value. The set of allowable filter names/values is dependent on the request being performed. Check the EC2 API guide for details.

Return type *list*

Returns A list of `boto.ec2.reservedinstance.ReservedInstancesOffering`

get_all_security_groups(*groupnames=None*, *group_ids=None*, *filters=None*)

Get all security groups associated with your account in a region.

Parameters

- **groupnames** (`list`) – A list of the names of security groups to retrieve. If not provided, all security groups will be returned.
- **group_ids** (`list`) – A list of IDs of security groups to retrieve for security groups within a VPC.
- **filters** (`dict`) – Optional filters that can be used to limit the results returned. Filters are provided in the form of a dictionary consisting of filter names as the key and filter values as the value. The set of allowable filter names/values is dependent on the request being performed. Check the EC2 API guide for details.

Return type *list*

Returns A list of `boto.ec2.securitygroup.SecurityGroup`

get_all_snapshots(*snapshot_ids=None*, *owner=None*, *restorable_by=None*, *filters=None*)

Get all EBS Snapshots associated with the current credentials.

Parameters

- **snapshot_ids** (`list`) – Optional list of snapshot ids. If this list is present, only the Snapshots associated with these snapshot ids will be returned.
- **owner** (*str*) – If present, only the snapshots owned by the specified user will be returned. Valid values are:
  - self
  - amazon

    – AWS Account ID

- **`restorable_by`** (`str`) – If present, only the snapshots that are restorable by the specified account id will be returned.
- **`filters`** (`dict`) – Optional filters that can be used to limit the results returned. Filters are provided in the form of a dictionary consisting of filter names as the key and filter values as the value. The set of allowable filter names/values is dependent on the request being performed. Check the EC2 API guide for details.

Return type list of *`boto.ec2.snapshot.Snapshot`*

Returns The requested Snapshot objects

`get_all_spot_instance_requests`(*request_ids=None*, *filters=None*)

Retrieve all the spot instances requests associated with your account.

Parameters

- **`request_ids`** (`list`) – A list of strings of spot instance request IDs
- **`filters`** (`dict`) – Optional filters that can be used to limit the results returned. Filters are provided in the form of a dictionary consisting of filter names as the key and filter values as the value. The set of allowable filter names/values is dependent on the request being performed. Check the EC2 API guide for details.

Return type *list*

Returns A list of `boto.ec2.spotinstancerequest.SpotInstanceRequest`

`get_all_tags`(*filters=None*)

Retrieve all the metadata tags associated with your account.

Parameters **`filters`** (`dict`) – Optional filters that can be used to limit the results returned. Filters are provided in the form of a dictionary consisting of filter names as the key and filter values as the value. The set of allowable filter names/values is dependent on the request being performed. Check the EC2 API guide for details.

Return type *dict*

Returns A dictionary containing metadata tags

`get_all_volumes`(*volume_ids=None*, *filters=None*)

Get all Volumes associated with the current credentials.

Parameters

- **`volume_ids`** (`list`) – Optional list of volume ids. If this list is present, only the volumes associated with these volume ids will be returned.
- **`filters`** (`dict`) – Optional filters that can be used to limit the results returned. Filters are provided in the form of a dictionary consisting of filter names as the key and filter values as the value. The set of allowable filter names/values is dependent on the request being performed. Check the EC2 API guide for details.

Return type list of *`boto.ec2.volume.Volume`*

Returns The requested Volume objects

`get_all_zones`(*zones=None*, *filters=None*)

Get all Availability Zones associated with the current region.

Parameters

- **zones** (list) – Optional list of zones. If this list is present, only the Zones associated with these zone names will be returned.
- **filters** (dict) – Optional filters that can be used to limit the results returned. Filters are provided in the form of a dictionary consisting of filter names as the key and filter values as the value. The set of allowable filter names/values is dependent on the request being performed. Check the EC2 API guide for details.

Return type list of *boto.ec2.zone.Zone*

Returns The requested Zone objects

get_console_output (*instance_id*)
Retrieves the console output for the specified instance.

Parameters **instance_id** (*string*) – The instance ID of a running instance on the cloud.

Return type *boto.ec2.instance.ConsoleOutput*

Returns The console output as a ConsoleOutput object

get_image (*image_id*)
Shortcut method to retrieve a specific image (AMI).

Parameters **image_id** (*string*) – the ID of the Image to retrieve

Return type *boto.ec2.image.Image*

Returns The EC2 Image specified or None if the image is not found

get_image_attribute (*image_id*, *attribute='launchPermission'*)
Gets an attribute from an image.

Parameters

- **image_id** (*string*) – The Amazon image id for which you want info about
- **attribute** (*string*) – The attribute you need information about. Valid choices are: * launchPermission * productCodes * blockDeviceMapping

Return type *boto.ec2.image.ImageAttribute*

Returns An ImageAttribute object representing the value of the attribute requested

get_instance_attribute (*instance_id*, *attribute*)
Gets an attribute from an instance.

Parameters

- **instance_id** (*string*) – The Amazon id of the instance
- **attribute** (*string*) – The attribute you need information about Valid choices are:
  - instanceType|kernel|ramdisk|userData|
  - disableApiTermination|
  - instanceInitiatedShutdownBehavior|
  - rootDeviceName|blockDeviceMapping

Return type `boto.ec2.image.InstanceAttribute`

Returns An InstanceAttribute object representing the value of the attribute requested

get_key_pair (*keyname*)
Convenience method to retrieve a specific keypair (KeyPair).

Parameters **image_id** (*string*) – the ID of the Image to retrieve

Return type *boto.ec2.keypair.KeyPair*

Returns The KeyPair specified or None if it is not found

get_params()
Returns a dictionary containing the value of of all of the keyword arguments passed when constructing this connection.

get_password_data(*instance_id*)
Get encrypted administrator password for a Windows instance.

Parameters **instance_id** (*string*) – The identifier of the instance to retrieve the password for.

get_snapshot_attribute(*snapshot_id*, *attribute='createVolumePermission'*)
Get information about an attribute of a snapshot. Only one attribute can be specified per call.

Parameters

- **snapshot_id** (*str*) – The ID of the snapshot.
- **attribute** (*str*) – The requested attribute. Valid values are:
  - createVolumePermission

Return type list of `boto.ec2.snapshotattribute.SnapshotAttribute`

Returns The requested Snapshot attribute

get_spot_datafeed_subscription()
Return the current spot instance data feed subscription associated with this account, if any.

Return type `boto.ec2.spotdatafeedsubscription.SpotDatafeedSubscription`

Returns The datafeed subscription object or None

get_spot_price_history(*start_time=None*, *end_time=None*, *instance_type=None*, *product_description=None*, *availability_zone=None*)
Retrieve the recent history of spot instances pricing.

Parameters

- **start_time** (*str*) – An indication of how far back to provide price changes for. An ISO8601 DateTime string.
- **end_time** (*str*) – An indication of how far forward to provide price changes for. An ISO8601 DateTime string.
- **instance_type** (*str*) – Filter responses to a particular instance type.
- **product_description** (*str*) – Filter responses to a particular platform. Valid values are currently: "Linux/UNIX", "SUSE Linux", and "Windows"
- **availability_zone** (*str*) – The availability zone for which prices should be returned

Return type *list*

Returns A list tuples containing price and timestamp.

import_key_pair(*key_name*, *public_key_material*)
mports the public key from an RSA key pair that you created with a third-party tool.

Supported formats:

•OpenSSH public key format (e.g., the format in ~/.ssh/authorized_keys)

•Base64 encoded DER format

•SSH public key file format as specified in RFC4716

DSA keys are not supported. Make sure your key generator is set up to create RSA keys.

Supported lengths: 1024, 2048, and 4096.

Parameters

- **`key_name`** (*`string`*) – The name of the new keypair
- **`public_key_material`** (*`string`*) – The public key. You must base64 encode the public key material before sending it to AWS.

Return type *`boto.ec2.keypair.KeyPair`*

Returns The newly created *`boto.ec2.keypair.KeyPair`*. The material attribute of the new KeyPair object will contain the the unencrypted PEM encoded RSA private key.

`modify_image_attribute`(*image_id*, *attribute='launchPermission'*, *operation='add'*, *user_ids=None*, *groups=None*, *product_codes=None*)
Changes an attribute of an image.

Parameters

- **`image_id`** (*`string`*) – The image id you wish to change
- **`attribute`** (*`string`*) – The attribute you wish to change
- **`operation`** (*`string`*) – Either add or remove (this is required for changing launchPermissions)
- **`user_ids`** (`list`) – The Amazon IDs of users to add/remove attributes
- **`groups`** (`list`) – The groups to add/remove attributes
- **`product_codes`** (`list`) – Amazon DevPay product code. Currently only one product code can be associated with an AMI. Once set, the product code cannot be changed or reset.

`modify_instance_attribute`(*instance_id*, *attribute*, *value*)
Changes an attribute of an instance

Parameters

- **`instance_id`** (*`string`*) – The instance id you wish to change
- **`attribute`** (*`string`*) – The attribute you wish to change.
  - AttributeName - Expected value (default)
  - instanceType - A valid instance type (m1.small)
  - kernel - Kernel ID (None)
  - ramdisk - Ramdisk ID (None)
  - userData - Base64 encoded String (None)
  - disableApiTermination - Boolean (true)
  - instanceInitiatedShutdownBehavior - stop|terminate
  - rootDeviceName - device name (None)
- **`value`** (*`string`*) – The new value for the attribute

Return type bool

Returns Whether the operation succeeded or not

modify_snapshot_attribute(*snapshot_id*, *attribute='createVolumePermission'*, *operation='add'*, *user_ids=None*, *groups=None*)

Changes an attribute of an image.

Parameters

- **snapshot_id** (*string*) – The snapshot id you wish to change
- **attribute** (*string*) – The attribute you wish to change. Valid values are: createVolumePermission
- **operation** (*string*) – Either add or remove (this is required for changing snapshot ermissions)
- **user_ids** (list) – The Amazon IDs of users to add/remove attributes
- **groups** (list) – The groups to add/remove attributes. The only valid value at this time is 'all'.

monitor_instance(*instance_id*)

Deprecated Version, maintained for backward compatibility. Enable CloudWatch monitoring for the supplied instance.

Parameters **instance_id** (*string*) – The instance id

Return type *list*

Returns A list of *boto.ec2.instanceinfo.InstanceInfo*

monitor_instances(*instance_ids*)

Enable CloudWatch monitoring for the supplied instances.

Parameters **instance_id** (*list of strings*) – The instance ids

Return type *list*

Returns A list of *boto.ec2.instanceinfo.InstanceInfo*

purchase_reserved_instance_offering(*reserved_instances_offering_id*, *instance_count=1*)

Purchase a Reserved Instance for use with your account. ** CAUTION ** This request can result in large amounts of money being charged to your AWS account. Use with caution!

Parameters

- **reserved_instances_offering_id** (*string*) – The offering ID of the Reserved Instance to purchase
- **instance_count** (*int*) – The number of Reserved Instances to purchase. Default value is 1.

Return type *boto.ec2.reservedinstance.ReservedInstance*

Returns The newly created Reserved Instance

reboot_instances(*instance_ids=None*)

Reboot the specified instances.

Parameters **instance_ids** (list) – The instances to terminate and reboot

register_image(*name=None*, *description=None*, *image_location=None*, *architecture=None*, *kernel_id=None*, *ramdisk_id=None*, *root_device_name=None*, *block_device_map=None*)

Register an image.

Parameters

- **name** (*string*) – The name of the AMI. Valid only for EBS-based images.
- **description** (*string*) – The description of the AMI.
- **image_location** (*string*) – Full path to your AMI manifest in Amazon S3 storage. Only used for S3-based AMI's.
- **architecture** (*string*) – The architecture of the AMI. Valid choices are: i386 | x86_64
- **kernel_id** (*string*) – The ID of the kernel with which to launch the instances
- **root_device_name** (*string*) – The root device name (e.g. /dev/sdh)
- **block_device_map** (boto.ec2.blockdevicemapping.BlockDeviceMapping) – A BlockDeviceMapping data structure describing the EBS volumes associated with the Image.

Return type string

Returns The new image id

release_address(*public_ip=None*, *allocation_id=None*)

Free up an Elastic IP address.

Parameters

- **public_ip** (*string*) – The public IP address for EC2 elastic IPs.
- **allocation_id** (*string*) – The ID for VPC elastic IPs.

Return type bool

Returns True if successful

request_spot_instances(*price*, *image_id*, *count=1*, *type='one-time'*, *valid_from=None*, *valid_until=None*, *launch_group=None*, *availability_zone_group=None*, *key_name=None*, *security_groups=None*, *user_data=None*, *addressing_type=None*, *instance_type='m1.small'*, *placement=None*, *kernel_id=None*, *ramdisk_id=None*, *monitoring_enabled=False*, *subnet_id=None*, *block_device_map=None*)

Request instances on the spot market at a particular price.

Parameters

- **price** (*str*) – The maximum price of your bid
- **image_id** (*string*) – The ID of the image to run
- **count** (*int*) – The of instances to requested
- **type** (*str*) – Type of request. Can be 'one-time' or 'persistent'. Default is one-time.
- **valid_from** (*str*) – Start date of the request. An ISO8601 time string.
- **valid_until** (*str*) – End date of the request. An ISO8601 time string.
- **launch_group** (*str*) – If supplied, all requests will be fulfilled as a group.

- **availability_zone_group** (*str*) – If supplied, all requests will be fulfilled within a single availability zone.
- **key_name** (*string*) – The name of the key pair with which to launch instances
- **security_groups** (*list of strings*) – The names of the security groups with which to associate instances
- **user_data** (*string*) – The user data passed to the launched instances
- **instance_type** (*string*) – The type of instance to run:
  - m1.small
  - m1.large
  - m1.xlarge
  - c1.medium
  - c1.xlarge
  - m2.xlarge
  - m2.2xlarge
  - m2.4xlarge
  - cc1.4xlarge
  - t1.micro
- **placement** (*string*) – The availability zone in which to launch the instances
- **kernel_id** (*string*) – The ID of the kernel with which to launch the instances
- **ramdisk_id** (*string*) – The ID of the RAM disk with which to launch the instances
- **monitoring_enabled** (*bool*) – Enable CloudWatch monitoring on the instance.
- **subnet_id** (*string*) – The subnet ID within which to launch the instances for VPC.
- **block_device_map** (`boto.ec2.blockdevicemapping.BlockDeviceMapping`) – A BlockDeviceMapping data structure describing the EBS volumes associated with the Image.

Return type *Reservation*

Returns The `boto.ec2.spotinstancerequest.SpotInstanceRequest` associated with the request for machines

reset_image_attribute (*image_id*, *attribute='launchPermission'*)
Resets an attribute of an AMI to its default value.

Parameters

- **image_id** (*string*) – ID of the AMI for which an attribute will be described
- **attribute** (*string*) – The attribute to reset

Return type bool

Returns Whether the operation succeeded or not

reset_instance_attribute (*instance_id*, *attribute*)
Resets an attribute of an instance to its default value.

Parameters

- **`instance_id`** (*string*) – ID of the instance
- **`attribute`** (*string*) – The attribute to reset. Valid values are: kernel|ramdisk

Return type bool

Returns Whether the operation succeeded or not

`reset_snapshot_attribute`(*snapshot_id*, *attribute='createVolumePermission'*)

Resets an attribute of a snapshot to its default value.

Parameters

- **`snapshot_id`** (*string*) – ID of the snapshot
- **`attribute`** (*string*) – The attribute to reset

Return type bool

Returns Whether the operation succeeded or not

`revoke_security_group`(*group_name=None*, *src_security_group_name=None*, *src_security_group_owner_id=None*, *ip_protocol=None*, *from_port=None*, *to_port=None*, *cidr_ip=None*, *group_id=None*, *src_security_group_group_id=None*)

Remove an existing rule from an existing security group. You need to pass in either src_security_group_name and src_security_group_owner_id OR ip_protocol, from_port, to_port, and cidr_ip. In other words, either you are revoking another group or you are revoking some ip-based rule.

Parameters

- **`group_name`** (*string*) – The name of the security group you are removing the rule from.
- **`src_security_group_name`** (*string*) – The name of the security group you are revoking access to.
- **`src_security_group_owner_id`** (*string*) – The ID of the owner of the security group you are revoking access to.
- **`ip_protocol`** (*string*) – Either tcp | udp | icmp
- **`from_port`** (*int*) – The beginning port number you are disabling
- **`to_port`** (*int*) – The ending port number you are disabling
- **`cidr_ip`** (*string*) – The CIDR block you are revoking access to. See http://goo.gl/Yj5QC

Return type bool

Returns True if successful.

`revoke_security_group_deprecated`(*group_name*, *src_security_group_name=None*, *src_security_group_owner_id=None*, *ip_protocol=None*, *from_port=None*, *to_port=None*, *cidr_ip=None*)

NOTE: This method uses the old-style request parameters that did not allow a port to be specified when authorizing a group.

Remove an existing rule from an existing security group. You need to pass in either src_security_group_name and src_security_group_owner_id OR ip_protocol, from_port, to_port, and cidr_ip. In other words, either you are revoking another group or you are revoking some ip-based rule.

Parameters

- **group_name** (*string*) – The name of the security group you are removing the rule from.
- **src_security_group_name** (*string*) – The name of the security group you are revoking access to.
- **src_security_group_owner_id** (*string*) – The ID of the owner of the security group you are revoking access to.
- **ip_protocol** (*string*) – Either tcp | udp | icmp
- **from_port** (*int*) – The beginning port number you are disabling
- **to_port** (*string*) – The ending port number you are disabling
- **to_port** – The CIDR block you are revoking access to. http://goo.gl/Yj5QC
- **group_id** (*string*) – ID of the EC2 or VPC security group to modify. This is required for VPC security groups and can be used instead of group_name for EC2 security groups.
- **group_id** – ID of the EC2 or VPC source security group. This is required for VPC security groups and can be used instead of group_name for EC2 security groups.

Return type bool

Returns True if successful.

revoke_security_group_egress(*group_id*, *ip_protocol*, *from_port=None*, *to_port=None*, *src_group_id=None*, *cidr_ip=None*)

Remove an existing egress rule from an existing VPC security group. You need to pass in an ip_protocol, from_port and to_port range only if the protocol you are using is port-based. You also need to pass in either a src_group_id or cidr_ip.

Parameters

- **group_id** – The name of the security group you are removing the rule from.
- **ip_protocol** (*string*) – Either tcp | udp | icmp | -1
- **from_port** (*int*) – The beginning port number you are disabling
- **to_port** (*int*) – The ending port number you are disabling
- **src_group_id** (*src_group_id*) – The source security group you are revoking access to.
- **cidr_ip** (*string*) – The CIDR block you are revoking access to. See http://goo.gl/Yj5QC

Return type bool

Returns True if successful.

run_instances(*image_id*, *min_count=1*, *max_count=1*, *key_name=None*, *security_groups=None*, *user_data=None*, *addressing_type=None*, *instance_type='m1.small'*, *placement=None*, *kernel_id=None*, *ramdisk_id=None*, *monitoring_enabled=False*, *subnet_id=None*, *block_device_map=None*, *disable_api_termination=False*, *instance_initiated_shutdown_behavior=None*, *private_ip_address=None*, *placement_group=None*, *client_token=None*, *security_group_ids=None*)

Runs an image on EC2.

Parameters

- **image_id** (*string*) – The ID of the image to run
- **min_count** (*int*) – The minimum number of instances to launch

- **max_count** (*int*) – The maximum number of instances to launch
- **key_name** (*string*) – The name of the key pair with which to launch instances
- **security_groups** (*list of strings*) – The names of the security groups with which to associate instances
- **user_data** (*string*) – The user data passed to the launched instances
- **instance_type** (*string*) – The type of instance to run:
  - m1.small
  - m1.large
  - m1.xlarge
  - c1.medium
  - c1.xlarge
  - m2.xlarge
  - m2.2xlarge
  - m2.4xlarge
  - cc1.4xlarge
  - t1.micro
- **placement** (*string*) – The availability zone in which to launch the instances
- **kernel_id** (*string*) – The ID of the kernel with which to launch the instances
- **ramdisk_id** (*string*) – The ID of the RAM disk with which to launch the instances
- **monitoring_enabled** (*bool*) – Enable CloudWatch monitoring on the instance.
- **subnet_id** (*string*) – The subnet ID within which to launch the instances for VPC.
- **private_ip_address** (*string*) – If you're using VPC, you can optionally use this parameter to assign the instance a specific available IP address from the subnet (e.g., 10.0.0.25).
- **block_device_map** (`boto.ec2.blockdevicemapping.BlockDeviceMapping`) – A BlockDeviceMapping data structure describing the EBS volumes associated with the Image.
- **disable_api_termination** (*bool*) – If True, the instances will be locked and will not be able to be terminated via the API.
- **instance_initiated_shutdown_behavior** (*string*) – Specifies whether the instance stops or terminates on instance-initiated shutdown. Valid values are:
  - stop
  - terminate
- **placement_group** (*string*) – If specified, this is the name of the placement group in which the instance(s) will be launched.
- **client_token** (*string*) – Unique, case-sensitive identifier you provide to ensure idempotency of the request. Maximum 64 ASCII characters
- **security_group_ids** (*list of strings*) – The ID of the VPC security groups with which to associate instances

Return type *Reservation*

Returns The `boto.ec2.instance.Reservation` associated with the request for machines

start_instances(*instance_ids=None*)
Start the instances specified

Parameters **instance_ids** (`list`) – A list of strings of the Instance IDs to start

Return type *list*

Returns A list of the instances started

stop_instances(*instance_ids=None*, *force=False*)
Stop the instances specified

Parameters

- **instance_ids** (`list`) – A list of strings of the Instance IDs to stop
- **force** (`bool`) – Forces the instance to stop

Return type *list*

Returns A list of the instances stopped

terminate_instances(*instance_ids=None*)
Terminate the instances specified

Parameters **instance_ids** (`list`) – A list of strings of the Instance IDs to terminate

Return type *list*

Returns A list of the instances terminated

trim_snapshots(*hourly_backups=8*, *daily_backups=7*, *weekly_backups=4*)
Trim excess snapshots, based on when they were taken. More current snapshots are retained, with the number retained decreasing as you move back in time.

If ebs volumes have a 'Name' tag with a value, their snapshots will be assigned the same tag when they are created. The values of the 'Name' tags for snapshots are used by this function to group snapshots taken from the same volume (or from a series of like-named volumes over time) for trimming.

For every group of like-named snapshots, this function retains the newest and oldest snapshots, as well as, by default, the first snapshots taken in each of the last eight hours, the first snapshots taken in each of the last seven days, the first snapshots taken in the last 4 weeks (counting Midnight Sunday morning as the start of the week), and the first snapshot from the first Sunday of each month forever.

Parameters

- **hourly_backups** (`int`) – How many recent hourly backups should be saved.
- **daily_backups** (`int`) – How many recent daily backups should be saved.
- **weekly_backups** (`int`) – How many recent weekly backups should be saved.

unmonitor_instance(*instance_id*)
Deprecated Version, maintained for backward compatibility. Disable CloudWatch monitoring for the supplied instance.

Parameters **instance_id** (`string`) – The instance id

Return type *list*

Returns A list of `boto.ec2.instanceinfo.InstanceInfo`

unmonitor_instances(*instance_ids*)
Disable CloudWatch monitoring for the supplied instance.

Parameters **instance_id** (*list of string*) – The instance id

Return type *list*

Returns A list of *boto.ec2.instanceinfo.InstanceInfo*

boto.ec2.ec2object

Represents an EC2 Object

class boto.ec2.ec2object.**EC2Object**(*connection=None*)

endElement(*name*, *value*, *connection*)

startElement(*name*, *attrs*, *connection*)

class boto.ec2.ec2object.**TaggedEC2Object**(*connection=None*)
Any EC2 resource that can be tagged should be represented by a Python object that subclasses this class. This class has the mechanism in place to handle the tagSet element in the Describe* responses. If tags are found, it will create a TagSet object and allow it to parse and collect the tags into a dict that is stored in the "tags" attribute of the object.

add_tag(*key*, *value=''*)
Add a tag to this object. Tag's are stored by AWS and can be used to organize and filter resources. Adding a tag involves a round-trip to the EC2 service.

Parameters

- **key** (*str*) – The key or name of the tag being stored.
- **value** (*str*) – An optional value that can be stored with the tag. If you want only the tag name and no value, the value should be the empty string.

remove_tag(*key*, *value=None*)
Remove a tag from this object. Removing a tag involves a round-trip to the EC2 service.

Parameters

- **key** (*str*) – The key or name of the tag being stored.
- **value** (*str*) – An optional value that can be stored with the tag. If a value is provided, it must match the value currently stored in EC2. If not, the tag will not be removed. If a value of None is provided, all tags with the specified name will be deleted. NOTE: There is an important distinction between a value of '' and a value of None.

startElement(*name*, *attrs*, *connection*)

boto.ec2.elb

See the *ELB Reference*.

boto.ec2.image

class boto.ec2.image.**Image**(*connection=None*)
Represents an EC2 Image

deregister(*delete_snapshot=False*)

endElement(*name*, *value*, *connection*)

get_kernel()

get_launch_permissions()

get_ramdisk()

remove_launch_permissions(*user_ids=None*, *group_names=None*)

reset_launch_attributes()

run(*min_count=1*, *max_count=1*, *key_name=None*, *security_groups=None*, *user_data=None*, *addressing_type=None*, *instance_type='m1.small'*, *placement=None*, *kernel_id=None*, *ramdisk_id=None*, *monitoring_enabled=False*, *subnet_id=None*, *block_device_map=None*, *disable_api_termination=False*, *instance_initiated_shutdown_behavior=None*, *private_ip_address=None*, *placement_group=None*, *security_group_ids=None*)

Runs this instance.

Parameters

- **min_count** (*int*) – The minimum number of instances to start
- **max_count** (*int*) – The maximum number of instances to start
- **key_name** (*string*) – The name of the keypair to run this instance with.
- **security_groups** –
- **user_data** –
- **daddressing_type** –
- **instance_type** (*string*) – The type of instance to run. Current choices are: m1.small | m1.large | m1.xlarge | c1.medium | c1.xlarge | m2.xlarge | m2.2xlarge | m2.4xlarge | cc1.4xlarge
- **placement** (*string*) – The availability zone in which to launch the instances
- **kernel_id** (*string*) – The ID of the kernel with which to launch the instances
- **ramdisk_id** (*string*) – The ID of the RAM disk with which to launch the instances
- **monitoring_enabled** (*bool*) – Enable CloudWatch monitoring on the instance.
- **subnet_id** (*string*) – The subnet ID within which to launch the instances for VPC.
- **private_ip_address** (*string*) – If you're using VPC, you can optionally use this parameter to assign the instance a specific available IP address from the subnet (e.g., 10.0.0.25).
- **block_device_map** (`boto.ec2.blockdevicemapping.BlockDeviceMapping`) – A BlockDeviceMapping data structure describing the EBS volumes associated with the Image.
- **disable_api_termination** (*bool*) – If True, the instances will be locked and will not be able to be terminated via the API.
- **instance_initiated_shutdown_behavior** (*string*) – Specifies whether the instance stops or terminates on instance-initiated shutdown. Valid values are: stop | terminate
- **placement_group** (*string*) – If specified, this is the name of the placement group in which the instance(s) will be launched.

- **`security_group_ids`** –

Return type *Reservation*

Returns The `boto.ec2.instance.Reservation` associated with the request for machines

`set_launch_permissions`(*user_ids=None*, *group_names=None*)

`startElement`(*name*, *attrs*, *connection*)

`update`(*validate=False*)
Update the image's state information by making a call to fetch the current image attributes from the service.

Parameters **`validate`** (`bool`) – By default, if EC2 returns no data about the image the update method returns quietly. If the validate param is True, however, it will raise a ValueError exception if no data is returned from EC2.

class `boto.ec2.image.`**`ImageAttribute`**(*parent=None*)

`endElement`(*name*, *value*, *connection*)

`startElement`(*name*, *attrs*, *connection*)

class `boto.ec2.image.`**`ProductCodes`**

`endElement`(*name*, *value*, *connection*)

`startElement`(*name*, *attrs*, *connection*)

boto.ec2.instance

Represents an EC2 Instance

class `boto.ec2.instance.`**`ConsoleOutput`**(*parent=None*)

`endElement`(*name*, *value*, *connection*)

`startElement`(*name*, *attrs*, *connection*)

class `boto.ec2.instance.`**`Group`**(*parent=None*)

`endElement`(*name*, *value*, *connection*)

`startElement`(*name*, *attrs*, *connection*)

class `boto.ec2.instance.`**`Instance`**(*connection=None*)

`confirm_product`(*product_code*)

`endElement`(*name*, *value*, *connection*)

`get_attribute`(*attribute*)
Gets an attribute from this instance.

Parameters **`attribute`** (`string`) – The attribute you need information about Valid choices are: instanceType|kernel|ramdisk|userData| disableApiTermination| instanceInitiatedShutdownBehavior| rootDeviceName|blockDeviceMapping

Return type `boto.ec2.image.InstanceAttribute`

Returns An InstanceAttribute object representing the value of the attribute requested

get_console_output()
Retrieves the console output for the instance.

Return type *boto.ec2.instance.ConsoleOutput*

Returns The console output as a ConsoleOutput object

modify_attribute(*attribute*, *value*)
Changes an attribute of this instance

Parameters

- **attribute** (*string*) – The attribute you wish to change. AttributeName - Expected value (default) instanceType - A valid instance type (m1.small) kernel - Kernel ID (None) ramdisk - Ramdisk ID (None) userData - Base64 encoded String (None) disableApiTermination - Boolean (true) instanceInitiatedShutdownBehavior - stop|terminate rootDeviceName - device name (None)
- **value** (*string*) – The new value for the attribute

Return type bool

Returns Whether the operation succeeded or not

monitor()

reboot()

reset_attribute(*attribute*)
Resets an attribute of this instance to its default value.

Parameters **attribute** (*string*) – The attribute to reset. Valid values are: kernel|ramdisk

Return type bool

Returns Whether the operation succeeded or not

start()
Start the instance.

startElement(*name*, *attrs*, *connection*)

stop(*force=False*)
Stop the instance

Parameters **force** (*bool*) – Forces the instance to stop

Return type *list*

Returns A list of the instances stopped

terminate()
Terminate the instance

unmonitor()

update(*validate=False*)
Update the instance's state information by making a call to fetch the current instance attributes from the service.

Parameters **validate** (*bool*) – By default, if EC2 returns no data about the instance the update method returns quietly. If the validate param is True, however, it will raise a ValueError exception if no data is returned from EC2.

use_ip(*ip_address*)

class `boto.ec2.instance.`**`InstanceAttribute`**(*parent=None*)

`ValidValues` **= ['instanceType', 'kernel', 'ramdisk', 'userData', 'disableApiTermination', 'instanceInitiatedShutdownB**

`endElement`(*name*, *value*, *connection*)

`startElement`(*name*, *attrs*, *connection*)

class `boto.ec2.instance.`**`Reservation`**(*connection=None*)

Represents a Reservation response object.

Variables

- ***`id`*** – The unique ID of the Reservation.
- **`owner_id`** – The unique ID of the owner of the Reservation.
- ***`groups`*** – A list of Group objects representing the security groups associated with launched instances.
- ***`instances`*** – A list of Instance objects launched in this Reservation.

`endElement`(*name*, *value*, *connection*)

`startElement`(*name*, *attrs*, *connection*)

`stop_all`()

class `boto.ec2.instance.`**`SubParse`**(*section*, *parent=None*)

`endElement`(*name*, *value*, *connection*)

`startElement`(*name*, *attrs*, *connection*)

boto.ec2.instanceinfo

class `boto.ec2.instanceinfo.`**`InstanceInfo`**(*connection=None*, *id=None*, *state=None*)

Represents an EC2 Instance status response from CloudWatch

Variables

- ***`id`*** (*`str`*) – The instance's EC2 ID.
- **`state`** (*`str`*) – Specifies the current status of the instance.

`endElement`(*name*, *value*, *connection*)

`startElement`(*name*, *attrs*, *connection*)

boto.ec2.instancestatus

class `boto.ec2.instancestatus.`**`Details`**

A dict object that contains name/value pairs which provide more detailed information about the status of the system or the instance.

`endElement`(*name*, *value*, *connection*)

`startElement`(*name*, *attrs*, *connection*)

class `boto.ec2.instancestatus.`**`Event`**(*code=None*, *description=None*, *not_before=None*, *not_after=None*)

A status event for an instance.

Variables

- `code` – A string indicating the event type.
- *`description`* – A string describing the reason for the event.
- **`not_before`** – A datestring describing the earliest time for the event.
- **`not_after`** – A datestring describing the latest time for the event.

`endElement`(*name*, *value*, *connection*)

`startElement`(*name*, *attrs*, *connection*)

class `boto.ec2.instancestatus.`**`EventSet`**

`endElement`(*name*, *value*, *connection*)

`startElement`(*name*, *attrs*, *connection*)

class `boto.ec2.instancestatus.`**`InstanceStatus`**(*id=None*, *zone=None*, *events=None*, *state_code=None*, *state_name=None*)

Represents an EC2 Instance status as reported by DescribeInstanceStatus request.

Variables

- *`id`* – The instance identifier.
- *`zone`* – The availability zone of the instance.
- **`events`** – A list of events relevant to the instance.
- **`state_code`** – An integer representing the current state of the instance.
- **`state_name`** – A string describing the current state of the instance.
- **`system_status`** – A Status object that reports impaired functionality that stems from issues related to the systems that support an instance, such as such as hardware failures and network connectivity problems.
- **`instance_status`** – A Status object that reports impaired functionality that arises from problems internal to the instance.

`endElement`(*name*, *value*, *connection*)

`startElement`(*name*, *attrs*, *connection*)

class `boto.ec2.instancestatus.`**`InstanceStatusSet`**(*connection=None*)

A list object that contains the results of a call to DescribeInstanceStatus request. Each element of the list will be an InstanceStatus object.

Variables *`next_token`* – If the response was truncated by the EC2 service, the next_token attribute of the object will contain the string that needs to be passed in to the next request to retrieve the next set of results.

`endElement`(*name*, *value*, *connection*)

`startElement`(*name*, *attrs*, *connection*)

class `boto.ec2.instancestatus.`**`Status`**(*status=None*, *details=None*)

A generic Status object used for system status and instance status.

Variables

- *`status`* – A string indicating overall status.

  - **details** – A dict containing name-value pairs which provide more details about the current status.

endElement (*name*, *value*, *connection*)

startElement (*name*, *attrs*, *connection*)

boto.ec2.keypair

Represents an EC2 Keypair

class boto.ec2.keypair.**KeyPair** (*connection=None*)

copy_to_region (*region*)
Create a new key pair of the same new in another region. Note that the new key pair will use a different ssh cert than the this key pair. After doing the copy, you will need to save the material associated with the new key pair (use the save method) to a local file.

Parameters **region** (boto.ec2.regioninfo.RegionInfo) – The region to which this security group will be copied.

Return type *boto.ec2.keypair.KeyPair*

Returns The new key pair

delete ()
Delete the KeyPair.

Return type bool

Returns True if successful, otherwise False.

endElement (*name*, *value*, *connection*)

save (*directory_path*)
Save the material (the unencrypted PEM encoded RSA private key) of a newly created KeyPair to a local file.

Parameters **directory_path** (*string*) – The fully qualified path to the directory in which the keypair will be saved. The keypair file will be named using the name of the keypair as the base name and .pem for the file extension. If a file of that name already exists in the directory, an exception will be raised and the old file will not be overwritten.

Return type bool

Returns True if successful.

boto.ec2.regioninfo

class boto.ec2.regioninfo.**EC2RegionInfo** (*connection=None*, *name=None*, *endpoint=None*)
Represents an EC2 Region

boto.ec2.reservedinstance

class `boto.ec2.reservedinstance.`**`ReservedInstance`**(*connection=None*, *id=None*, *instance_type=None*, *availability_zone=None*, *duration=None*, *fixed_price=None*, *usage_price=None*, *description=None*, *instance_count=None*, *state=None*)

`endElement`(*name*, *value*, *connection*)

class `boto.ec2.reservedinstance.`**`ReservedInstancesOffering`**(*connection=None*, *id=None*, *instance_type=None*, *availability_zone=None*, *duration=None*, *fixed_price=None*, *usage_price=None*, *description=None*)

`describe`()

`endElement`(*name*, *value*, *connection*)

`purchase`(*instance_count=1*)

`startElement`(*name*, *attrs*, *connection*)

boto.ec2.securitygroup

Represents an EC2 Security Group

class `boto.ec2.securitygroup.`**`GroupOrCIDR`**(*parent=None*)

`endElement`(*name*, *value*, *connection*)

`startElement`(*name*, *attrs*, *connection*)

class `boto.ec2.securitygroup.`**`IPPermissions`**(*parent=None*)

`add_grant`(*name=None*, *owner_id=None*, *cidr_ip=None*)

`endElement`(*name*, *value*, *connection*)

`startElement`(*name*, *attrs*, *connection*)

class `boto.ec2.securitygroup.`**`IPPermissionsList`**

`endElement`(*name*, *value*, *connection*)

`startElement`(*name*, *attrs*, *connection*)

class `boto.ec2.securitygroup.`**`SecurityGroup`**(*connection=None*, *owner_id=None*, *name=None*, *description=None*, *id=None*)

`add_rule`(*ip_protocol*, *from_port*, *to_port*, *src_group_name*, *src_group_owner_id*, *cidr_ip*)

Add a rule to the SecurityGroup object. Note that this method only changes the local version of the object. No information is sent to EC2.

authorize (*ip_protocol=None*, *from_port=None*, *to_port=None*, *cidr_ip=None*, *src_group=None*)
Add a new rule to this security group. You need to pass in either src_group_name OR ip_protocol, from_port, to_port, and cidr_ip. In other words, either you are authorizing another group or you are authorizing some ip-based rule.

Parameters

- **ip_protocol** (*string*) – Either tcp | udp | icmp
- **from_port** (*int*) – The beginning port number you are enabling
- **to_port** (*int*) – The ending port number you are enabling
- **cidr_ip** (*string*) – The CIDR block you are providing access to. See http://en.wikipedia.org/wiki/Classless_Inter-Domain_Routing
- **src_group** (*boto.ec2.securitygroup.SecurityGroup* or *boto.ec2.securitygroup.GroupOrCIDR*) – The Security Group you are granting access to.

Return type bool

Returns True if successful.

copy_to_region (*region*, *name=None*)
Create a copy of this security group in another region. Note that the new security group will be a separate entity and will not stay in sync automatically after the copy operation.

Parameters

- **region** (boto.ec2.regioninfo.RegionInfo) – The region to which this security group will be copied.
- **name** (*string*) – The name of the copy. If not supplied, the copy will have the same name as this security group.

Return type *boto.ec2.securitygroup.SecurityGroup*

Returns The new security group.

delete ()

endElement (*name*, *value*, *connection*)

instances ()
Find all of the current instances that are running within this security group.

Return type list of *boto.ec2.instance.Instance*

Returns A list of Instance objects

remove_rule (*ip_protocol*, *from_port*, *to_port*, *src_group_name*, *src_group_owner_id*, *cidr_ip*)
Remove a rule to the SecurityGroup object. Note that this method only changes the local version of the object. No information is sent to EC2.

revoke (*ip_protocol=None*, *from_port=None*, *to_port=None*, *cidr_ip=None*, *src_group=None*)

startElement (*name*, *attrs*, *connection*)

boto.ec2.snapshot

Represents an EC2 Elastic Block Store Snapshot

class boto.ec2.snapshot.**Snapshot** (*connection=None*)

AttrName = 'createVolumePermission'

delete()

endElement(*name*, *value*, *connection*)

get_permissions()

reset_permissions()

share(*user_ids=None*, *groups=None*)

unshare(*user_ids=None*, *groups=None*)

update(*validate=False*)
Update the data associated with this snapshot by querying EC2.

Parameters **validate** (*bool*) – By default, if EC2 returns no data about the snapshot the update method returns quietly. If the validate param is True, however, it will raise a ValueError exception if no data is returned from EC2.

class boto.ec2.snapshot.**SnapshotAttribute**(*parent=None*)

endElement(*name*, *value*, *connection*)

startElement(*name*, *attrs*, *connection*)

boto.ec2.volume

Represents an EC2 Elastic Block Storage Volume

class boto.ec2.volume.**AttachmentSet**

endElement(*name*, *value*, *connection*)

startElement(*name*, *attrs*, *connection*)

class boto.ec2.volume.**Volume**(*connection=None*)

attach(*instance_id*, *device*)
Attach this EBS volume to an EC2 instance.

Parameters

- **instance_id** (*str*) – The ID of the EC2 instance to which it will be attached.
- **device** (*str*) – The device on the instance through which the volume will be exposed (e.g. /dev/sdh)

Return type bool

Returns True if successful

attachment_state()
Get the attachment state.

create_snapshot(*description=None*)
Create a snapshot of this EBS Volume.

Parameters **description** (*str*) – A description of the snapshot. Limited to 256 characters.

Return type *boto.ec2.snapshot.Snapshot*

Returns The created Snapshot object

delete()
Delete this EBS volume.

Return type bool

Returns True if successful

detach(*force=False*)
Detach this EBS volume from an EC2 instance.

Parameters **force** (*bool*) – Forces detachment if the previous detachment attempt did not occur cleanly. This option can lead to data loss or a corrupted file system. Use this option only as a last resort to detach a volume from a failed instance. The instance will not have an opportunity to flush file system caches nor file system meta data. If you use this option, you must perform file system check and repair procedures.

Return type bool

Returns True if successful

endElement(*name*, *value*, *connection*)

snapshots(*owner=None*, *restorable_by=None*)
Get all snapshots related to this volume. Note that this requires that all available snapshots for the account be retrieved from EC2 first and then the list is filtered client-side to contain only those for this volume.

Parameters

- **owner** (*str*) – If present, only the snapshots owned by the specified user will be returned. Valid values are: self | amazon | AWS Account ID
- **restorable_by** (*str*) – If present, only the snapshots that are restorable by the specified account id will be returned.

Return type list of L{boto.ec2.snapshot.Snapshot}

Returns The requested Snapshot objects

startElement(*name*, *attrs*, *connection*)

update(*validate=False*)
Update the data associated with this volume by querying EC2.

Parameters **validate** (*bool*) – By default, if EC2 returns no data about the volume the update method returns quietly. If the validate param is True, however, it will raise a ValueError exception if no data is returned from EC2.

volume_state()
Returns the state of the volume. Same value as the status attribute.

boto.ec2.zone

Represents an EC2 Availability Zone

class boto.ec2.zone.**MessageSet**
A list object that contains messages associated with an availability zone.

endElement(*name*, *value*, *connection*)

startElement(*name*, *attrs*, *connection*)

class `boto.ec2.zone.`**`Zone`**(*connection=None*)

Represents an Availability Zone.

Variables

- ***`name`*** – The name of the zone.
- **`state`** – The current state of the zone.
- ***`region_name`*** – The name of the region the zone is associated with.
- **`messages`** – A list of messages related to the zone.

`endElement`(*name*, *value*, *connection*)

`startElement`(*name*, *attrs*, *connection*)

An Introduction to boto's Elastic Mapreduce interface

This tutorial focuses on the boto interface to Elastic Mapreduce from Amazon Web Services. This tutorial assumes that you have already downloaded and installed boto.

Creating a Connection

The first step in accessing Elastic Mapreduce is to create a connection to the service. There are two ways to do this in boto. The first is:

```
>>> from boto.emr.connection import EmrConnection
>>> conn = EmrConnection('<aws access key>', '<aws secret key>')
```

At this point the variable conn will point to an EmrConnection object. In this example, the AWS access key and AWS secret key are passed in to the method explicitly. Alternatively, you can set the environment variables:

AWS_ACCESS_KEY_ID - Your AWS Access Key ID AWS_SECRET_ACCESS_KEY - Your AWS Secret Access Key

and then call the constructor without any arguments, like this:

```
>>> conn = EmrConnection()
```

There is also a shortcut function in the boto package called connect_emr that may provide a slightly easier means of creating a connection:

```
>>> import boto
>>> conn = boto.connect_emr()
```

In either case, conn points to an EmrConnection object which we will use throughout the remainder of this tutorial.

Creating Streaming JobFlow Steps

Upon creating a connection to Elastic Mapreduce you will next want to create one or more jobflow steps. There are two types of steps, streaming and custom jar, both of which have a class in the boto Elastic Mapreduce implementation.

Creating a streaming step that runs the AWS wordcount example, itself written in Python, can be accomplished by:

```
>>> from boto.emr.step import StreamingStep
>>> step = StreamingStep(name='My wordcount example',
...                      mapper='s3n://elasticmapreduce/samples/wordcount/
↪wordSplitter.py',
...                      reducer='aggregate',
...                      input='s3n://elasticmapreduce/samples/wordcount/input',
...                      output='s3n://<my output bucket>/output/wordcount_output')
```

where <my output bucket> is a bucket you have created in S3.

Note that this statement does not run the step, that is accomplished later when we create a jobflow.

Additional arguments of note to the streaming jobflow step are cache_files, cache_archive and step_args. The options cache_files and cache_archive enable you to use the Hadoops distributed cache to share files amongst the instances that run the step. The argument step_args allows one to pass additional arguments to Hadoop streaming, for example modifications to the Hadoop job configuration.

Creating Custom Jar Job Flow Steps

The second type of jobflow step executes tasks written with a custom jar. Creating a custom jar step for the AWS CloudBurst example can be accomplished by:

```
>>> from boto.emr.step import JarStep
>>> step = JarStep(name='Coudburst example',
...                jar='s3n://elasticmapreduce/samples/cloudburst/cloudburst.jar',
...                step_args=['s3n://elasticmapreduce/samples/cloudburst/input/s_suis.
↪br',
...                           's3n://elasticmapreduce/samples/cloudburst/input/100k.br
↪',
...                           's3n://<my output bucket>/output/cloudfront_output',
...                            36, 3, 0, 1, 240, 48, 24, 24, 128, 16])
```

Note that this statement does not actually run the step, that is accomplished later when we create a jobflow. Also note that this JarStep does not include a main_class argument since the jar MANIFEST.MF has a Main-Class entry.

Creating JobFlows

Once you have created one or more jobflow steps, you will next want to create and run a jobflow. Creating a jobflow that executes either of the steps we created above can be accomplished by:

```
>>> import boto
>>> conn = boto.connect_emr()
>>> jobid = conn.run_jobflow(name='My jobflow',
...                          log_uri='s3://<my log uri>/jobflow_logs',
...                          steps=[step])
```

The method will not block for the completion of the jobflow, but will immediately return. The status of the jobflow can be determined by:

```
>>> status = conn.describe_jobflow(jobid)
>>> status.state
u'STARTING'
```

One can then use this state to block for a jobflow to complete. Valid jobflow states currently defined in the AWS API are COMPLETED, FAILED, TERMINATED, RUNNING, SHUTTING_DOWN, STARTING and WAITING.

In some cases you may not have built all of the steps prior to running the jobflow. In these cases additional steps can be added to a jobflow by running:

```
>>> conn.add_jobflow_steps(jobid, [second_step])
```

If you wish to add additional steps to a running jobflow you may want to set the keep_alive parameter to True in run_jobflow so that the jobflow does not automatically terminate when the first step completes.

The run_jobflow method has a number of important parameters that are worth investigating. They include parameters to change the number and type of EC2 instances on which the jobflow is executed, set a SSH key for manual debugging and enable AWS console debugging.

Terminating JobFlows

By default when all the steps of a jobflow have finished or failed the jobflow terminates. However, if you set the keep_alive parameter to True or just want to halt the execution of a jobflow early you can terminate a jobflow by:

```
>>> import boto
>>> conn = boto.connect_emr()
>>> conn.terminate_jobflow('<jobflow id>')
```

EMR

boto.emr

This module provies an interface to the Elastic MapReduce (EMR) service from AWS.

boto.emr.connection

Represents a connection to the EMR service

class `boto.emr.connection.`**`EmrConnection`**(*aws_access_key_id=None, aws_secret_access_key=None, is_secure=True, port=None, proxy=None, proxy_port=None, proxy_user=None, proxy_pass=None, debug=0, https_connection_factory=None, region=None, path='/'*)

`APIVersion` = **'2009-03-31'**

`DebuggingArgs` = **'s3n://us-east-1.elasticmapreduce/libs/state-pusher/0.1/fetch'**

`DebuggingJar` = **'s3n://us-east-1.elasticmapreduce/libs/script-runner/script-runner.jar'**

`DefaultRegionEndpoint` = **'elasticmapreduce.amazonaws.com'**

`DefaultRegionName` = **'us-east-1'**

`ResponseError`
alias of `EmrResponseError`

`add_instance_groups`(*jobflow_id, instance_groups*)
Adds instance groups to a running cluster.

Parameters

- **jobflow_id** (*str*) – The id of the jobflow which will take the new instance groups
- **instance_groups** (list *(boto.emr.InstanceGroup)*) – A list of instance groups to add to the job

add_jobflow_steps (*jobflow_id*, *steps*)
Adds steps to a jobflow

Parameters

- **jobflow_id** (*str*) – The job flow id
- **steps** (list *(boto.emr.Step)*) – A list of steps to add to the job

describe_jobflow (*jobflow_id*)
Describes a single Elastic MapReduce job flow

Parameters **jobflow_id** (*str*) – The job flow id of interest

describe_jobflows (*states=None*, *jobflow_ids=None*, *created_after=None*, *created_before=None*)
Retrieve all the Elastic MapReduce job flows on your account

Parameters

- **states** (list) – A list of strings with job flow states wanted
- **jobflow_ids** (list) – A list of job flow IDs
- **created_after** (*datetime*) – Bound on job flow creation time
- **created_before** (*datetime*) – Bound on job flow creation time

modify_instance_groups (*instance_group_ids*, *new_sizes*)
Modify the number of nodes and configuration settings in an instance group.

Parameters

- **instance_group_ids** (list *(str)*) – A list of the ID's of the instance groups to be modified
- **new_sizes** (list *(int)*) – A list of the new sizes for each instance group

run_jobflow (*name*, *log_uri*, *ec2_keyname=None*, *availability_zone=None*, *master_instance_type='m1.small'*, *slave_instance_type='m1.small'*, *num_instances=1*, *action_on_failure='TERMINATE_JOB_FLOW'*, *keep_alive=False*, *enable_debugging=False*, *hadoop_version=None*, *steps=[]*, *bootstrap_actions=[]*, *instance_groups=None*, *additional_info=None*, *ami_version='1.0'*, *api_params=None*)
Runs a job flow :type name: str :param name: Name of the job flow

Parameters

- **log_uri** (*str*) – URI of the S3 bucket to place logs
- **ec2_keyname** (*str*) – EC2 key used for the instances
- **availability_zone** (*str*) – EC2 availability zone of the cluster
- **master_instance_type** (*str*) – EC2 instance type of the master
- **slave_instance_type** (*str*) – EC2 instance type of the slave nodes
- **num_instances** (*int*) – Number of instances in the Hadoop cluster
- **action_on_failure** (*str*) – Action to take if a step terminates
- **keep_alive** (*bool*) – Denotes whether the cluster should stay alive upon completion

- **enable_debugging** (*bool*) – Denotes whether AWS console debugging should be enabled.
- **hadoop_version** (*str*) – Version of Hadoop to use. If ami_version is not set, defaults to '0.20' for backwards compatibility with older versions of boto.
- **steps** (list *(boto.emr.Step)*) – List of steps to add with the job
- **bootstrap_actions** (list *(boto.emr.BootstrapAction)*) – List of bootstrap actions that run before Hadoop starts.
- **instance_groups** (list *(boto.emr.InstanceGroup)*) – Optional list of instance groups to use when creating this job. NB: When provided, this argument supersedes num_instances and master/slave_instance_type.
- **ami_version** (*str*) – Amazon Machine Image (AMI) version to use for instances. Values accepted by EMR are '1.0', '2.0', and 'latest'; EMR currently defaults to '1.0' if you don't set 'ami_version'.
- **additional_info** (*JSON str*) – A JSON string for selecting additional features
- **api_params** (dict) – a dictionary of additional parameters to pass directly to the EMR API (so you don't have to upgrade boto to use new EMR features). You can also delete an API parameter by setting it to None.

Return type str

Returns The jobflow id

set_termination_protection(*jobflow_id*, *termination_protection_status*)
Set termination protection on specified Elastic MapReduce job flows

Parameters

- **jobflow_ids** (list *or* *str*) – A list of job flow IDs
- **termination_protection_status** (*bool*) – Termination protection status

terminate_jobflow(*jobflow_id*)
Terminate an Elastic MapReduce job flow

Parameters **jobflow_id** (*str*) – A jobflow id

terminate_jobflows(*jobflow_ids*)
Terminate an Elastic MapReduce job flow

Parameters **jobflow_ids** (list) – A list of job flow IDs

boto.emr.step

class boto.emr.step.**JarStep**(*name*, *jar*, *main_class=None*, *action_on_failure='TERMINATE_JOB_FLOW'*, *step_args=None*)
Custom jar step

A elastic mapreduce step that executes a jar

Parameters

- **name** (*str*) – The name of the step
- **jar** (*str*) – S3 URI to the Jar file
- **main_class** (*str*) – The class to execute in the jar
- **action_on_failure** (*str*) – An action, defined in the EMR docs to take on failure.

- **step_args** (list (*str*)) – A list of arguments to pass to the step

args()

jar()

main_class()

class boto.emr.step.**Step**
Jobflow Step base class

args()

Return type *list*(str)

Returns List of arguments for the step

jar()

Return type str

Returns URI to the jar

main_class()

Return type str

Returns The main class name

class boto.emr.step.**StreamingStep**(*name, mapper, reducer=None, combiner=None, action_on_failure='TERMINATE_JOB_FLOW', cache_files=None, cache_archives=None, step_args=None, input=None, output=None, jar='/home/hadoop/contrib/streaming/hadoop-streaming.jar'*)
Hadoop streaming step

A hadoop streaming elastic mapreduce step

Parameters

- **name** (*str*) – The name of the step
- **mapper** (*str*) – The mapper URI
- **reducer** (*str*) – The reducer URI
- **combiner** (*str*) – The combiner URI. Only works for Hadoop 0.20 and later!
- **action_on_failure** (*str*) – An action, defined in the EMR docs to take on failure.
- **cache_files** (list (*str*)) – A list of cache files to be bundled with the job
- **cache_archives** (list (*str*)) – A list of jar archives to be bundled with the job
- **step_args** (list (*str*)) – A list of arguments to pass to the step
- **input** (*str or a list of str*) – The input uri
- **output** (*str*) – The output uri
- **jar** (*str*) – The hadoop streaming jar. This can be either a local path on the master node, or an s3:// URI.

args()

jar()

main_class()

boto.emr.emrobject

This module contains EMR response objects

class `boto.emr.emrobject.`**`AddInstanceGroupsResponse`**(*connection=None*)

> **`Fields`** **= set(['InstanceGroupIds', 'JobFlowId'])**

class `boto.emr.emrobject.`**`Arg`**(*connection=None*)

> **`endElement`**(*name*, *value*, *connection*)

class `boto.emr.emrobject.`**`BootstrapAction`**(*connection=None*)

> **`Fields`** **= set(['Path', 'Args', 'Name'])**
>
> **`startElement`**(*name*, *attrs*, *connection*)

class `boto.emr.emrobject.`**`EmrObject`**(*connection=None*)

> **`Fields`** **= set([])**
>
> **`endElement`**(*name*, *value*, *connection*)
>
> **`startElement`**(*name*, *attrs*, *connection*)

class `boto.emr.emrobject.`**`InstanceGroup`**(*connection=None*)

> **`Fields`** **= set(['ReadyDateTime', 'InstanceType', 'InstanceRole', 'EndDateTime', 'InstanceRunningCount', 'State', 'Bid**

class `boto.emr.emrobject.`**`JobFlow`**(*connection=None*)

> **`Fields`** **= set(['TerminationProtected', 'MasterInstanceId', 'State', 'HadoopVersion', 'LogUri', 'AmiVersion', 'Ec2KeyN**
>
> **`startElement`**(*name*, *attrs*, *connection*)

class `boto.emr.emrobject.`**`KeyValue`**(*connection=None*)

> **`Fields`** **= set(['Value', 'Key'])**

class `boto.emr.emrobject.`**`ModifyInstanceGroupsResponse`**(*connection=None*)

> **`Fields`** **= set(['RequestId'])**

class `boto.emr.emrobject.`**`RunJobFlowResponse`**(*connection=None*)

> **`Fields`** **= set(['JobFlowId'])**

class `boto.emr.emrobject.`**`Step`**(*connection=None*)

> **`Fields`** **= set(['Name', 'EndDateTime', 'Jar', 'ActionOnFailure', 'State', 'MainClass', 'StartDateTime', 'CreationDateTi**
>
> **`startElement`**(*name*, *attrs*, *connection*)

An Introduction to boto's Autoscale interface

This tutorial focuses on the boto interface to the Autoscale service. This assumes you are familiar with boto's EC2 interface and concepts.

Autoscale Concepts

The AWS Autoscale service is comprised of three core concepts:

1. *Autoscale Group (AG):* An AG can be viewed as a collection of criteria for maintaining or scaling a set of EC2 instances over one or more availability zones. An AG is limited to a single region.
2. *Launch Configuration (LC):* An LC is the set of information needed by the AG to launch new instances - this can encompass image ids, startup data, security groups and keys. Only one LC is attached to an AG.
3. *Triggers*: A trigger is essentially a set of rules for determining when to scale an AG up or down. These rules can encompass a set of metrics such as average CPU usage across instances, or incoming requests, a threshold for when an action will take place, as well as parameters to control how long to wait after a threshold is crossed.

Creating a Connection

The first step in accessing autoscaling is to create a connection to the service. There are two ways to do this in boto. The first is:

```
>>> from boto.ec2.autoscale import AutoScaleConnection
>>> conn = AutoScaleConnection('<aws access key>', '<aws secret key>')
```

Alternatively, you can use the shortcut:

```
>>> conn = boto.connect_autoscale()
```

A Note About Regions and Endpoints

Like EC2 the Autoscale service has a different endpoint for each region. By default the US endpoint is used. To choose a specific region, instantiate the AutoScaleConnection object with that region's endpoint.

```
>>> ec2 = boto.connect_autoscale(host='autoscaling.eu-west-1.amazonaws.com')
```

Alternatively, edit your boto.cfg with the default Autoscale endpoint to use:

```
[Boto]
autoscale_endpoint = autoscaling.eu-west-1.amazonaws.com
```

Getting Existing AutoScale Groups

To retrieve existing autoscale groups:

```
>>> conn.get_all_groups()
```

You will get back a list of AutoScale group objects, one for each AG you have.

Creating Autoscaling Groups

An Autoscaling group has a number of parameters associated with it.

1. *Name*: The name of the AG.
2. *Availability Zones*: The list of availability zones it is defined over.
3. *Minimum Size*: Minimum number of instances running at one time.
4. *Maximum Size*: Maximum number of instances running at one time.
5. *Launch Configuration (LC)*: A set of instructions on how to launch an instance.
6. *Load Balancer*: An optional ELB load balancer to use. See the ELB tutorial for information on how to create a load balancer.

For the purposes of this tutorial, let's assume we want to create one autoscale group over the us-east-1a and us-east-1b availability zones. We want to have two instances in each availability zone, thus a minimum size of 4. For now we won't worry about scaling up or down - we'll introduce that later when we talk about triggers. Thus we'll set a maximum size of 4 as well. We'll also associate the AG with a load balancer which we assume we've already created, called 'my_lb'.

Our LC tells us how to start an instance. This will at least include the image id to use, security_group, and key information. We assume the image id, key name and security groups have already been defined elsewhere - see the EC2 tutorial for information on how to create these.

```
>>> from boto.ec2.autoscale import LaunchConfiguration
>>> from boto.ec2.autoscale import AutoScalingGroup
>>> lc = LaunchConfiguration(name='my-launch_config', image_id='my-ami',
                             key_name='my_key_name',
                             security_groups=['my_security_groups'])
>>> conn.create_launch_configuration(lc)
```

We now have created a launch configuration called 'my-launch-config'. We are now ready to associate it with our new autoscale group.

```
>>> ag = AutoScalingGroup(group_name='my_group', load_balancers=['my-lb'],
                          availability_zones=['us-east-1a', 'us-east-1b'],
                          launch_config=lc, min_size=4, max_size=4)
>>> conn.create_auto_scaling_group(ag)
```

We now have a new autoscaling group defined! At this point instances should be starting to launch. To view activity on an autoscale group:

```
>>> ag.get_activities()
 [Activity:Launching a new EC2 instance status:Successful progress:100,
  ...]
```

or alternatively:

```
>>> conn.get_all_activities(ag)
```

This autoscale group is fairly useful in that it will maintain the minimum size without breaching the maximum size defined. That means if one instance crashes, the autoscale group will use the launch configuration to start a new one in an attempt to maintain its minimum defined size. It knows instance health using the health check defined on its associated load balancer.

Scaling a Group Up or Down

It might be more useful to also define means to scale a group up or down depending on certain criteria. For example, if the average CPU utilization of all your instances goes above 60%, you may want to scale up a number of instances to deal with demand - likewise you might want to scale down if usage drops. These criteria are defined in *triggers*.

For example, let's modify our above group to have a maxsize of 8 and define means of scaling up based on CPU utilization. We'll say we should scale up if the average CPU usage goes above 80% and scale down if it goes below 40%.

```
>>> from boto.ec2.autoscale import Trigger
>>> tr = Trigger(name='my_trigger', autoscale_group=ag,
             measure_name='CPUUtilization', statistic='Average',
             unit='Percent',
             dimensions=[('AutoScalingGroupName', ag.name)],
             period=60, lower_threshold=40,
             lower_breach_scale_increment='-5',
             upper_threshold=80,
             upper_breach_scale_increment='10',
             breach_duration=360)
>> conn.create_trigger(tr)
```

Auto Scaling Reference

boto.ec2.autoscale

This module provides an interface to the Elastic Compute Cloud (EC2) Auto Scaling service.

class `boto.ec2.autoscale.`**`AutoScaleConnection`**(*aws_access_key_id=None*, *aws_secret_access_key=None*, *is_secure=True*, *port=None*, *proxy=None*, *proxy_port=None*, *proxy_user=None*, *proxy_pass=None*, *debug=0*, *https_connection_factory=None*, *region=None*, *path='/'*)

Init method to create a new connection to the AutoScaling service.

B{Note:} The host argument is overridden by the host specified in the boto configuration file.

`APIVersion` = '2011-01-01'

`DefaultRegionEndpoint` = 'autoscaling.amazonaws.com'

`DefaultRegionName` = 'us-east-1'

`build_list_params`(*params*, *items*, *label*)

Items is a list of dictionaries or strings:

```
[
    {
        'Protocol' : 'HTTP',
        'LoadBalancerPort' : '80',
        'InstancePort' : '80'
    },
    ..
] etc.
```

or:

```
['us-east-1b',...]
```

create_auto_scaling_group(*as_group*)
Create auto scaling group.

create_launch_configuration(*launch_config*)
Creates a new Launch Configuration.

Parameters **launch_config** (*boto.ec2.autoscale.launchconfig.LaunchConfiguration*) – LaunchConfiguration object.

create_or_update_tags(*tags*)
Creates new tags or updates existing tags for an Auto Scaling group.

Parameters **tags** (List of `boto.ec2.autoscale.tag.Tag`) – The new or updated tags.

create_scaling_policy(*scaling_policy*)
Creates a new Scaling Policy.

Parameters **scaling_policy** (*boto.ec2.autoscale.policy.ScalingPolicy*) – ScalingPolicy object.

create_scheduled_group_action(*as_group*, *name*, *time*, *desired_capacity=None*, *min_size=None*, *max_size=None*)
Creates a scheduled scaling action for a Auto Scaling group. If you leave a parameter unspecified, the corresponding value remains unchanged in the affected Auto Scaling group.

Parameters

- **as_group** (*string*) – The auto scaling group to get activities on.
- **name** (*string*) – Scheduled action name.
- **time** (*datetime.datetime*) – The time for this action to start.
- **desired_capacity** (*int*) – The number of EC2 instances that should be running in this group.
- **min_size** (*int*) – The minimum size for the new auto scaling group.
- **max_size** (*int*) – The minimum size for the new auto scaling group.

delete_auto_scaling_group(*name*, *force_delete=False*)
Deletes the specified auto scaling group if the group has no instances and no scaling activities in progress.

delete_launch_configuration(*launch_config_name*)
Deletes the specified LaunchConfiguration.

The specified launch configuration must not be attached to an Auto Scaling group. Once this call completes, the launch configuration is no longer available for use.

delete_policy(*policy_name*, *autoscale_group=None*)
Delete a policy.

Parameters

- **policy_name** (*str*) – The name or ARN of the policy to delete.
- **autoscale_group** (*str*) – The name of the autoscale group.

delete_scheduled_action(*scheduled_action_name*, *autoscale_group=None*)
Deletes a previously scheduled action.

Parameters

- **scheduled_action_name** (*str*) – The name of the action you want to delete.
- **autoscale_group** (*str*) – The name of the autoscale group.

delete_tags(*tags*)

Deletes existing tags for an Auto Scaling group.

Parameters tags (List of `boto.ec2.autoscale.tag.Tag`) – The new or updated tags.

disable_metrics_collection(*as_group*, *metrics=None*)

Disables monitoring of group metrics for the Auto Scaling group specified in AutoScalingGroupName. You can specify the list of affected metrics with the Metrics parameter.

enable_metrics_collection(*as_group*, *granularity*, *metrics=None*)

Enables monitoring of group metrics for the Auto Scaling group specified in AutoScalingGroupName. You can specify the list of enabled metrics with the Metrics parameter.

Auto scaling metrics collection can be turned on only if the InstanceMonitoring.Enabled flag, in the Auto Scaling group's launch configuration, is set to true.

Parameters

- **autoscale_group** (*string*) – The auto scaling group to get activities on.
- **granularity** (*string*) – The granularity to associate with the metrics to collect. Currently, the only legal granularity is "1Minute".
- **metrics** (*string list*) – The list of metrics to collect. If no metrics are specified, all metrics are enabled.

execute_policy(*policy_name*, *as_group=None*, *honor_cooldown=None*)

get_all_activities(*autoscale_group*, *activity_ids=None*, *max_records=None*, *next_token=None*)

Get all activities for the given autoscaling group.

This action supports pagination by returning a token if there are more pages to retrieve. To get the next page, call this action again with the returned token as the NextToken parameter

Parameters

- **autoscale_group** (str or *boto.ec2.autoscale.group.AutoScalingGroup* object) – The auto scaling group to get activities on.
- **max_records** (*int*) – Maximum amount of activities to return.

Return type *list*

Returns List of *boto.ec2.autoscale.activity.Activity* instances.

get_all_adjustment_types()

get_all_autoscaling_instances(*instance_ids=None*, *max_records=None*, *next_token=None*)

Returns a description of each Auto Scaling instance in the instance_ids list. If a list is not provided, the service returns the full details of all instances up to a maximum of fifty.

This action supports pagination by returning a token if there are more pages to retrieve. To get the next page, call this action again with the returned token as the NextToken parameter.

Parameters

- **instance_ids** (*list*) – List of Autoscaling Instance IDs which should be searched for.
- **max_records** (*int*) – Maximum number of results to return.

Return type *list*

Returns List of `boto.ec2.autoscale.activity.Activity` objects.

get_all_groups(*names=None*, *max_records=None*, *next_token=None*)
Returns a full description of each Auto Scaling group in the given list. This includes all Amazon EC2 instances that are members of the group. If a list of names is not provided, the service returns the full details of all Auto Scaling groups.

This action supports pagination by returning a token if there are more pages to retrieve. To get the next page, call this action again with the returned token as the NextToken parameter.

Parameters

- **names** (`list`) – List of group names which should be searched for.
- **max_records** (*int*) – Maximum amount of groups to return.

Return type *list*

Returns List of `boto.ec2.autoscale.group.AutoScalingGroup` instances.

get_all_launch_configurations(***kwargs*)
Returns a full description of the launch configurations given the specified names.

If no names are specified, then the full details of all launch configurations are returned.

Parameters

- **names** (`list`) – List of configuration names which should be searched for.
- **max_records** (*int*) – Maximum amount of configurations to return.
- **next_token** (*str*) – If you have more results than can be returned at once, pass in this parameter to page through all results.

Return type *list*

Returns List of `boto.ec2.autoscale.launchconfig.LaunchConfiguration` instances.

get_all_metric_collection_types()
Returns a list of metrics and a corresponding list of granularities for each metric.

get_all_policies(*as_group=None*, *policy_names=None*, *max_records=None*, *next_token=None*)
Returns descriptions of what each policy does. This action supports pagination. If the response includes a token, there are more records available. To get the additional records, repeat the request with the response token as the NextToken parameter.

If no group name or list of policy names are provided, all available policies are returned.

Parameters

- **as_name** (*str*) – The name of the `boto.ec2.autoscale.group.AutoScalingGroup` to filter for.
- **names** (`list`) – List of policy names which should be searched for.
- **max_records** (*int*) – Maximum amount of groups to return.

get_all_scaling_process_types()
Returns scaling process types for use in the ResumeProcesses and SuspendProcesses actions.

get_all_scheduled_actions(*as_group=None*, *start_time=None*, *end_time=None*, *scheduled_actions=None*, *max_records=None*, *next_token=None*)

get_all_tags (*filters=None*, *max_records=None*, *next_token=None*)
Lists the Auto Scaling group tags.

This action supports pagination by returning a token if there are more pages to retrieve. To get the next page, call this action again with the returned token as the NextToken parameter.

Parameters

- **filters** (dict) – The value of the filter type used to identify the tags to be returned. NOT IMPLEMENTED YET.
- **max_records** (*int*) – Maximum number of tags to return.

Return type *list*

Returns List of `boto.ec2.autoscale.tag.Tag` instances.

resume_processes (*as_group*, *scaling_processes=None*)
Resumes Auto Scaling processes for an Auto Scaling group.

Parameters

- **as_group** (*string*) – The auto scaling group to resume processes on.
- **scaling_processes** (list) – Processes you want to resume. If omitted, all processes will be resumed.

set_instance_health (*instance_id*, *health_status*, *should_respect_grace_period=True*)
Explicitly set the health status of an instance.

Parameters

- **instance_id** (*str*) – The identifier of the EC2 instance.
- **health_status** (*str*) – The health status of the instance. "Healthy" means that the instance is healthy and should remain in service. "Unhealthy" means that the instance is unhealthy. Auto Scaling should terminate and replace it.
- **should_respect_grace_period** (*bool*) – If True, this call should respect the grace period associated with the group.

suspend_processes (*as_group*, *scaling_processes=None*)
Suspends Auto Scaling processes for an Auto Scaling group.

Parameters

- **as_group** (*string*) – The auto scaling group to suspend processes on.
- **scaling_processes** (list) – Processes you want to suspend. If omitted, all processes will be suspended.

terminate_instance (*instance_id*, *decrement_capacity=True*)
Terminates the specified instance. The desired group size can also be adjusted, if desired.

Parameters

- **instance_id** (*str*) – The ID of the instance to be terminated.
- **decrement_capacity** – Whether to decrement the size of the autoscaling group or not.

`boto.ec2.autoscale.`**connect_to_region** (*region_name*, ***kw_params*)
Given a valid region name, return a *boto.ec2.autoscale.AutoScaleConnection*.

Parameters **region_name** (*str*) – The name of the region to connect to.

Return type `boto.ec2.AutoScaleConnection` or `None`

Returns A connection to the given region, or None if an invalid region name is given

`boto.ec2.autoscale.`**`regions`**`()`

Get all available regions for the Auto Scaling service.

Return type *list*

Returns A list of `boto.RegionInfo` instances

boto.ec2.autoscale.activity

class `boto.ec2.autoscale.activity.`**`Activity`**`(`*connection=None*`)`

`endElement``(`*name*, *value*, *connection*`)`

`startElement``(`*name*, *attrs*, *connection*`)`

boto.ec2.autoscale.group

class `boto.ec2.autoscale.group.`**`AutoScalingGroup`**`(`*connection=None*, *name=None*, *launch_config=None*, *availability_zones=None*, *load_balancers=None*, *default_cooldown=None*, *health_check_type=None*, *health_check_period=None*, *placement_group=None*, *vpc_zone_identifier=None*, *desired_capacity=None*, *min_size=None*, *max_size=None*, ***kwargs*`)`

Creates a new AutoScalingGroup with the specified name.

You must not have already used up your entire quota of AutoScalingGroups in order for this call to be successful. Once the creation request is completed, the AutoScalingGroup is ready to be used in other calls.

Parameters

- **`name`** (*`str`*) – Name of autoscaling group (required).
- **`availability_zones`** (`list`) – List of availability zones (required).
- **`default_cooldown`** (*`int`*) – Number of seconds after a Scaling Activity completes before any further scaling activities can start.
- **`desired_capacity`** (*`int`*) – The desired capacity for the group.
- **`health_check_period`** (*`str`*) – Length of time in seconds after a new EC2 instance comes into service that Auto Scaling starts checking its health.
- **`health_check_type`** (*`str`*) – The service you want the health status from, Amazon EC2 or Elastic Load Balancer.
- **`launch_config`** (*`str or`* `LaunchConfiguration`) – Name of launch configuration (required).
- **`load_balancers`** (`list`) – List of load balancers.
- **`maxsize`** – Maximum size of group (required).
- **`minsize`** – Minimum size of group (required).

- **placement_group** (*str*) – Physical location of your cluster placement group created in Amazon EC2.
- **vpc_zone_identifier** (*str*) – The subnet identifier of the Virtual Private Cloud.

Return type *boto.ec2.autoscale.group.AutoScalingGroup*

Returns An autoscale group.

cooldown

delete(*force_delete=False*)
Delete this auto-scaling group if no instances attached or no scaling activities in progress.

endElement(*name*, *value*, *connection*)

get_activities(*activity_ids=None*, *max_records=50*)
Get all activies for this group.

resume_processes(*scaling_processes=None*)
Resumes Auto Scaling processes for an Auto Scaling group.

set_capacity(*capacity*)
Set the desired capacity for the group.

shutdown_instances()
Convenience method which shuts down all instances associated with this group.

startElement(*name*, *attrs*, *connection*)

suspend_processes(*scaling_processes=None*)
Suspends Auto Scaling processes for an Auto Scaling group.

update()
Sync local changes with AutoScaling group.

class boto.ec2.autoscale.group.**AutoScalingGroupMetric**(*connection=None*)

endElement(*name*, *value*, *connection*)

startElement(*name*, *attrs*, *connection*)

class boto.ec2.autoscale.group.**EnabledMetric**(*connection=None*, *metric=None*, *granularity=None*)

endElement(*name*, *value*, *connection*)

startElement(*name*, *attrs*, *connection*)

class boto.ec2.autoscale.group.**ProcessType**(*connection=None*)

endElement(*name*, *value*, *connection*)

startElement(*name*, *attrs*, *connection*)

class boto.ec2.autoscale.group.**SuspendedProcess**(*connection=None*)

endElement(*name*, *value*, *connection*)

startElement(*name*, *attrs*, *connection*)

boto.ec2.autoscale.instance

class `boto.ec2.autoscale.instance.`**`Instance`**(*connection=None*)

`endElement`(*name*, *value*, *connection*)

`startElement`(*name*, *attrs*, *connection*)

boto.ec2.autoscale.launchconfig

class `boto.ec2.autoscale.launchconfig.`**`BlockDeviceMapping`**(*connection=None*, *device_name=None*, *virtual_name=None*)

`endElement`(*name*, *value*, *connection*)

`startElement`(*name*, *attrs*, *connection*)

class `boto.ec2.autoscale.launchconfig.`**`Ebs`**(*connection=None*, *snapshot_id=None*, *volume_size=None*)

`endElement`(*name*, *value*, *connection*)

`startElement`(*name*, *attrs*, *connection*)

class `boto.ec2.autoscale.launchconfig.`**`InstanceMonitoring`**(*connection=None*, *enabled='false'*)

`endElement`(*name*, *value*, *connection*)

`startElement`(*name*, *attrs*, *connection*)

class `boto.ec2.autoscale.launchconfig.`**`LaunchConfiguration`**(*connection=None*, *name=None*, *image_id=None*, *key_name=None*, *security_groups=None*, *user_data=None*, *instance_type='m1.small'*, *kernel_id=None*, *ramdisk_id=None*, *block_device_mappings=None*, *instance_monitoring=False*)

A launch configuration.

Parameters

- **`name`** (*`str`*) – Name of the launch configuration to create.
- **`image_id`** (*`str`*) – Unique ID of the Amazon Machine Image (AMI) which was assigned during registration.
- **`key_name`** (*`str`*) – The name of the EC2 key pair.
- **`security_groups`** (`list`) – Names of the security groups with which to associate the EC2 instances.
- **`user_data`** (*`str`*) – The user data available to launched EC2 instances.
- **`instance_type`** (*`str`*) – The instance type

- **kern_id** (*str*) – Kernel id for instance
- **ramdisk_id** (*str*) – RAM disk id for instance
- **block_device_mappings** (list) – Specifies how block devices are exposed for instances
- **instance_monitoring** (*bool*) – Whether instances in group are launched with detailed monitoring.

delete()
Delete this launch configuration.

endElement(*name*, *value*, *connection*)

startElement(*name*, *attrs*, *connection*)

boto.ec2.autoscale.policy

class boto.ec2.autoscale.policy.**AdjustmentType**(*connection=None*)

endElement(*name*, *value*, *connection*)

startElement(*name*, *attrs*, *connection*)

class boto.ec2.autoscale.policy.**Alarm**(*connection=None*)

endElement(*name*, *value*, *connection*)

startElement(*name*, *attrs*, *connection*)

class boto.ec2.autoscale.policy.**MetricCollectionTypes**(*connection=None*)

class BaseType(*connection*)

arg = ‘’

endElement(*name*, *value*, *connection*)

startElement(*name*, *attrs*, *connection*)

class MetricCollectionTypes.**Granularity**(*connection*)

arg = ‘Granularity’

class MetricCollectionTypes.**Metric**(*connection*)

arg = ‘Metric’

MetricCollectionTypes.**endElement**(*name*, *value*, *connection*)

MetricCollectionTypes.**startElement**(*name*, *attrs*, *connection*)

class boto.ec2.autoscale.policy.**ScalingPolicy**(*connection=None*, ***kwargs*)
Scaling Policy

Parameters

- **name** (*str*) – Name of scaling policy.

- **adjustment_type** (*str*) – Specifies the type of adjustment. Valid values are *ChangeInCapacity*, *ExactCapacity* and *PercentChangeInCapacity*.
- **as_name** (*str or int*) – Name or ARN of the Auto Scaling Group.
- **scaling_adjustment** (*int*) – Value of adjustment (type specified in *adjustment_type*).
- **cooldown** (*int*) – Time (in seconds) before Alarm related Scaling Activities can start after the previous Scaling Activity ends.

delete()

endElement(*name*, *value*, *connection*)

startElement(*name*, *attrs*, *connection*)

boto.ec2.autoscale.request

class boto.ec2.autoscale.request.**Request**(*connection=None*)

endElement(*name*, *value*, *connection*)

startElement(*name*, *attrs*, *connection*)

boto.ec2.autoscale.scheduled

class boto.ec2.autoscale.scheduled.**ScheduledUpdateGroupAction**(*connection=None*)

endElement(*name*, *value*, *connection*)

startElement(*name*, *attrs*, *connection*)

CloudFront

This new boto module provides an interface to Amazon's Content Service, CloudFront.

Warning: This module is not well tested. Paging of distributions is not yet supported. CNAME support is completely untested. Use with caution. Feedback and bug reports are greatly appreciated.

Creating a CloudFront connection

```
>>> import boto
>>> c = boto.connect_cloudfront()
```

Create a new *boto.cloudfront.distribution.Distribution*:

```
>>> distro = c.create_distribution(origin='mybucket.s3.amazonaws.com', enabled=False,␣
↪comment='My new Distribution')
>>> d.domain_name
u'd2oxf3980lnb8l.cloudfront.net'
>>> d.id
```

```
u'ECH69MOIW7613'
>>> d.status
u'InProgress'
>>> d.config.comment
u'My new distribution'
>>> d.config.origin
<S3Origin: mybucket.s3.amazonaws.com>
>>> d.config.caller_reference
u'31b8d9cf-a623-4a28-b062-a91856fac6d0'
>>> d.config.enabled
False
```

Note that a new caller reference is created automatically, using uuid.uuid4(). The `boto.cloudfront.distribution.Distribution`, `boto.cloudfront.distribution.DistributionConfig` and `boto.cloudfront.distribution.DistributionSummary` objects are defined in the `boto.cloudfront.distribution` module.

To get a listing of all current distributions:

```
>>> rs = c.get_all_distributions()
>>> rs
[<boto.cloudfront.distribution.DistributionSummary instance at 0xe8d4e0>,
 <boto.cloudfront.distribution.DistributionSummary instance at 0xe8d788>]
```

This returns a list of `boto.cloudfront.distribution.DistributionSummary` objects. Note that paging is not yet supported! To get a `boto.cloudfront.distribution.DistributionObject` from a `boto.cloudfront.distribution.DistributionSummary` object:

```
>>> ds = rs[1]
>>> distro = ds.get_distribution()
>>> distro.domain_name
u'd2oxf3980lnb8l.cloudfront.net'
```

To change a property of a distribution object:

```
>>> distro.comment
u'My new distribution'
>>> distro.update(comment='This is a much better comment')
>>> distro.comment
'This is a much better comment'
```

You can also enable/disable a distribution using the following convenience methods:

```
>>> distro.enable()  # just calls distro.update(enabled=True)
```

or

```
>>> distro.disable()  # just calls distro.update(enabled=False)
```

The only attributes that can be updated for a Distribution are comment, enabled and cnames.

To delete a `boto.cloudfront.distribution.Distribution`:

```
>>> distro.delete()
```

CloudFront

boto.cloudfront

class `boto.cloudfront.`**`CloudFrontConnection`**(*aws_access_key_id=None, aws_secret_access_key=None, port=None, proxy=None, proxy_port=None, host='cloudfront.amazonaws.com', debug=0*)

> **`DefaultHost`** = **'cloudfront.amazonaws.com'**
>
> **`Version`** = **'2010-11-01'**
>
> **`create_distribution`**(*origin, enabled, caller_reference='', cnames=None, comment='', trusted_signers=None*)
>
> **`create_invalidation_request`**(*distribution_id, paths, caller_reference=None*)
> Creates a new invalidation request :see: http://goo.gl/8vECq
>
> **`create_origin_access_identity`**(*caller_reference='', comment=''*)
>
> **`create_streaming_distribution`**(*origin, enabled, caller_reference='', cnames=None, comment='', trusted_signers=None*)
>
> **`delete_distribution`**(*distribution_id, etag*)
>
> **`delete_origin_access_identity`**(*access_id, etag*)
>
> **`delete_streaming_distribution`**(*distribution_id, etag*)
>
> **`get_all_distributions`**()
>
> **`get_all_origin_access_identity`**()
>
> **`get_all_streaming_distributions`**()
>
> **`get_distribution_config`**(*distribution_id*)
>
> **`get_distribution_info`**(*distribution_id*)
>
> **`get_etag`**(*response*)
>
> **`get_origin_access_identity_config`**(*access_id*)
>
> **`get_origin_access_identity_info`**(*access_id*)
>
> **`get_streaming_distribution_config`**(*distribution_id*)
>
> **`get_streaming_distribution_info`**(*distribution_id*)
>
> **`invalidation_request_status`**(*distribution_id, request_id, caller_reference=None*)
>
> **`set_distribution_config`**(*distribution_id, etag, config*)
>
> **`set_origin_access_identity_config`**(*access_id, etag, config*)
>
> **`set_streaming_distribution_config`**(*distribution_id, etag, config*)

boto.cloudfront.distribution

class `boto.cloudfront.distribution.`**`Distribution`**(*connection=None, config=None, domain_name='', id='', last_modified_time=None, status=''*)

add_object(*name*, *content*, *headers=None*, *replace=True*)
Adds a new content object to the Distribution. The content for the object will be copied to a new Key in the S3 Bucket and the permissions will be set appropriately for the type of Distribution.

Parameters

- **name** (*str or unicode*) – The name or key of the new object.
- **content** (*file-like object*) – A file-like object that contains the content for the new object.
- **headers** (dict) – A dictionary containing additional headers you would like associated with the new object in S3.

Return type `boto.cloudfront.object.Object`

Returns The newly created object.

create_signed_url(*url*, *keypair_id*, *expire_time=None*, *valid_after_time=None*, *ip_address=None*, *policy_url=None*, *private_key_file=None*, *private_key_string=None*)
Creates a signed CloudFront URL that is only valid within the specified parameters.

Parameters

- **url** (*str*) – The URL of the protected object.
- **keypair_id** (*str*) – The keypair ID of the Amazon KeyPair used to sign theURL. This ID MUST correspond to the private key specified with private_key_file or private_key_string.
- **expire_time** (*int*) – The expiry time of the URL. If provided, the URL will expire after the time has passed. If not provided the URL will never expire. Format is a unix epoch. Use time.time() + duration_in_sec.
- **valid_after_time** (*int*) – If provided, the URL will not be valid until after valid_after_time. Format is a unix epoch. Use time.time() + secs_until_valid.
- **ip_address** (*str*) – If provided, only allows access from the specified IP address. Use '192.168.0.10' for a single IP or use '192.168.0.0/24' CIDR notation for a subnet.
- **policy_url** (*str*) – If provided, allows the signature to contain wildcard globs in the URL. For example, you could provide: 'http://example.com/media/*' and the policy and signature would allow access to all contents of the media subdirectory. If not specified, only allow access to the exact url provided in 'url'.
- **private_key_file** (*str or file object.*) – If provided, contains the filename of the private key file used for signing or an open file object containing the private key contents. Only one of private_key_file or private_key_string can be provided.
- **private_key_string** (*str*) – If provided, contains the private key string used for signing. Only one of private_key_file or private_key_string can be provided.

Return type str

Returns The signed URL.

delete()
Delete this CloudFront Distribution. The content associated with the Distribution is not deleted from the underlying Origin bucket in S3.

disable()
Activate the Distribution. A convenience wrapper around the update method.

enable()
Deactivate the Distribution. A convenience wrapper around the update method.

endElement(*name*, *value*, *connection*)

get_objects()

Return a list of all content objects in this distribution.

Return type list of `boto.cloudfront.object.Object`

Returns The content objects

set_permissions(*object*, *replace=False*)

Sets the S3 ACL grants for the given object to the appropriate value based on the type of Distribution. If the Distribution is serving private content the ACL will be set to include the Origin Access Identity associated with the Distribution. If the Distribution is serving public content the content will be set up with "public-read".

Parameters

- **enabled** – The Object whose ACL is being set
- **replace** (*bool*) – If False, the Origin Access Identity will be appended to the existing ACL for the object. If True, the ACL for the object will be completely replaced with one that grants READ permission to the Origin Access Identity.

set_permissions_all(*replace=False*)

Sets the S3 ACL grants for all objects in the Distribution to the appropriate value based on the type of Distribution.

Parameters **replace** (*bool*) – If False, the Origin Access Identity will be appended to the existing ACL for the object. If True, the ACL for the object will be completely replaced with one that grants READ permission to the Origin Access Identity.

startElement(*name*, *attrs*, *connection*)

update(*enabled=None*, *cnames=None*, *comment=None*)

Update the configuration of the Distribution. The only values of the DistributionConfig that can be directly updated are:

- CNAMES
- Comment
- Whether the Distribution is enabled or not

Any changes to the `trusted_signers` or `origin` properties of this distribution's current config object will also be included in the update. Therefore, to set the origin access identity for this distribution, set `Distribution.config.origin.origin_access_identity` before calling this update method.

Parameters

- **enabled** (*bool*) – Whether the Distribution is active or not.
- **cnames** (*list of str*) – The DNS CNAME's associated with this Distribution. Maximum of 10 values.
- **comment** (*str or unicode*) – The comment associated with the Distribution.

class `boto.cloudfront.distribution.`**DistributionConfig**(*connection=None*, *origin=None*, *enabled=False*, *caller_reference=''*, *cnames=None*, *comment=''*, *trusted_signers=None*, *default_root_object=None*, *logging=None*)

Parameters

- **origin** (*boto.cloudfront.origin.S3Origin* or *boto.cloudfront.origin.CustomOrigin*) – Origin information to associate with the distribution. If your distribution will use an Amazon S3 origin, then this should be an S3Origin object. If your distribution will use a custom origin (non Amazon S3), then this should be a CustomOrigin object.
- **enabled** (*array of str*) – Whether the distribution is enabled to accept end user requests for content.
- **caller_reference** – A unique number that ensures the request can't be replayed. If no caller_reference is provided, boto will generate a type 4 UUID for use as the caller reference.
- **cnames** – A CNAME alias you want to associate with this distribution. You can have up to 10 CNAME aliases per distribution.
- **comment** (*str*) – Any comments you want to include about the distribution.
- **trusted_signers** (*:class`boto.cloudfront.signers.TrustedSigners`*) – Specifies any AWS accounts you want to permit to create signed URLs for private content. If you want the distribution to use signed URLs, this should contain a TrustedSigners object; if you want the distribution to use basic URLs, leave this None.
- **default_root_object** – Designates a default root object. Only include a DefaultRootObject value if you are going to assign a default root object for the distribution.
- **logging** (*:class`boto.cloudfront.logging.LoggingInfo`*) – Controls whether access logs are written for the distribution. If you want to turn on access logs, this should contain a LoggingInfo object; otherwise it should contain None.

endElement (*name*, *value*, *connection*)

startElement (*name*, *attrs*, *connection*)

to_xml ()

class boto.cloudfront.distribution.**DistributionSummary** (*connection=None*, *domain_name=''*, *id=''*, *last_modified_time=None*, *status=''*, *origin=None*, *cname=''*, *comment=''*, *enabled=False*)

endElement (*name*, *value*, *connection*)

get_distribution ()

startElement (*name*, *attrs*, *connection*)

class boto.cloudfront.distribution.**StreamingDistribution** (*connection=None*, *config=None*, *domain_name=''*, *id=''*, *last_modified_time=None*, *status=''*)

delete ()

startElement (*name*, *attrs*, *connection*)

update (*enabled=None*, *cnames=None*, *comment=None*)
Update the configuration of the StreamingDistribution. The only values of the StreamingDistributionConfig that can be directly updated are:

•CNAMES

•Comment

•Whether the Distribution is enabled or not

Any changes to the `trusted_signers` or `origin` properties of this distribution's current config object will also be included in the update. Therefore, to set the origin access identity for this distribution, set `StreamingDistribution.config.origin.origin_access_identity` before calling this update method.

Parameters

- **enabled** (*bool*) – Whether the StreamingDistribution is active or not.
- **cnames** (*list of str*) – The DNS CNAME's associated with this Distribution. Maximum of 10 values.
- **comment** (*str or unicode*) – The comment associated with the Distribution.

class boto.cloudfront.distribution.**StreamingDistributionConfig** (*connection=None*, *origin=''*, *enabled=False*, *caller_reference=''*, *cnames=None*, *comment=''*, *trusted_signers=None*, *logging=None*)

to_xml ()

class boto.cloudfront.distribution.**StreamingDistributionSummary** (*connection=None*, *domain_name=''*, *id=''*, *last_modified_time=None*, *status=''*, *origin=None*, *cname=''*, *comment=''*, *enabled=False*)

get_distribution ()

boto.cloudfront.origin

class boto.cloudfront.origin.**CustomOrigin** (*dns_name=None*, *http_port=80*, *https_port=443*, *origin_protocol_policy=None*)
Origin information to associate with the distribution. If your distribution will use a non-Amazon S3 origin, then you use the CustomOrigin element.

Parameters

- **dns_name** (*str*) – The DNS name of your Amazon S3 bucket to associate with the distribution. For example: mybucket.s3.amazonaws.com.
- **http_port** (*int*) – The HTTP port the custom origin listens on.

- **`https_port`** – The HTTPS port the custom origin listens on.
- **`origin_protocol_policy`** (*str*) – The origin protocol policy to apply to your origin. If you specify http-only, CloudFront will use HTTP only to access the origin. If you specify match-viewer, CloudFront will fetch from your origin using HTTP or HTTPS, based on the protocol of the viewer request.

`endElement`(*name*, *value*, *connection*)

`startElement`(*name*, *attrs*, *connection*)

`to_xml`()

class `boto.cloudfront.origin.`**`S3Origin`**(*dns_name=None*, *origin_access_identity=None*)
Origin information to associate with the distribution. If your distribution will use an Amazon S3 origin, then you use the S3Origin element.

Parameters

- **`dns_name`** (*str*) – The DNS name of your Amazon S3 bucket to associate with the distribution. For example: mybucket.s3.amazonaws.com.
- **`origin_access_identity`** (*str*) – The CloudFront origin access identity to associate with the distribution. If you want the distribution to serve private content, include this element; if you want the distribution to serve public content, remove this element.

`endElement`(*name*, *value*, *connection*)

`startElement`(*name*, *attrs*, *connection*)

`to_xml`()

`boto.cloudfront.origin.`**`get_oai_value`**(*origin_access_identity*)

boto.cloudfront.exception

exception `boto.cloudfront.exception.`**`CloudFrontServerError`**(*status*, *reason*, *body=None*, **args*)

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.

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
>>> conn = boto.connect_sdb(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_items() 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_items(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_items() 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
>>>
```

SDB Reference

In addition to what is seen below, boto includes an abstraction layer for SimpleDB that may be used:

- *SimpleDB DB* (Maintained, but little documentation)

boto.sdb

`boto.sdb.`**`connect_to_region`**`(`*region_name*, ***kw_params*`)`
Given a valid region name, return a `boto.sdb.connection.SDBConnection`.

Type str

Parameters **`region_name`** – The name of the region to connect to.

Return type `boto.sdb.connection.SDBConnection` or `None`

Returns A connection to the given region, or None if an invalid region name is given

`boto.sdb.`**`get_region`**`(`*region_name*, ***kw_params*`)`
Find and return a `boto.sdb.regioninfo.RegionInfo` object given a region name.

Type str

Param The name of the region.

Return type `boto.sdb.regioninfo.RegionInfo`

Returns The RegionInfo object for the given region or None if an invalid region name is provided.

`boto.sdb.`**`regions`**`()`
Get all available regions for the SDB service.

Return type *list*

Returns A list of `boto.sdb.regioninfo.RegionInfo` instances

boto.sdb.connection

class `boto.sdb.connection.`**`ItemThread`**`(`*name*, *domain_name*, *item_names*`)`
A threaded `Item` retriever utility class. Retrieved `Item` objects are stored in the `items` instance variable after `run()` is called.

Tip: The item retrieval will not start until the `run()` method is called.

Parameters

- **`name`** (`str`) – A thread name. Used for identification.

- **domain_name** (*str*) – The name of a SimpleDB *Domain*
- **item_names** (*string or list of strings*) – The name(s) of the items to retrieve from the specified *Domain*.

Variables ***items*** (list) – A list of items retrieved. Starts as empty list.

run()
Start the threaded retrieval of items. Populates the `items` list with *Item* objects.

class boto.sdb.connection.**SDBConnection**(*aws_access_key_id=None, aws_secret_access_key=None, is_secure=True, port=None, proxy=None, proxy_port=None, proxy_user=None, proxy_pass=None, debug=0, https_connection_factory=None, region=None, path='/', converter=None, security_token=None*)
This class serves as a gateway to your SimpleDB region (defaults to us-east-1). Methods within allow access to SimpleDB *Domain* objects and their associated *Item* objects.

Tip: While you may instantiate this class directly, it may be easier to go through *boto.connect_sdb()*.

For any keywords that aren't documented, refer to the parent class, *boto.connection.AWSAuthConnection*. You can avoid having to worry about these keyword arguments by instantiating these objects via *boto.connect_sdb()*.

Parameters region (boto.sdb.regioninfo.SDBRegionInfo) – Explicitly specify a region. Defaults to `us-east-1` if not specified. You may also specify the region in your `boto.cfg`:

```
[SDB]
region = eu-west-1
```

APIVersion = '2009-04-15'

DefaultRegionEndpoint = 'sdb.amazonaws.com'

DefaultRegionName = 'us-east-1'

ResponseError
alias of `SDBResponseError`

batch_delete_attributes(*domain_or_name, items*)
Delete multiple items in a domain.

Parameters

- **domain_or_name** (string or *boto.sdb.domain.Domain* object.) – Either the name of a domain or a Domain object
- **items** (*dict or dict-like object*) – A dictionary-like object. The keys of the dictionary are the item names and the values are either:
  - dictionaries of attribute names/values, exactly the same as the attribute_names parameter of the scalar put_attributes call. The attribute name/value pairs will only be deleted if they match the name/value pairs passed in.
  - None which means that all attributes associated with the item should be deleted.

Returns True if successful

batch_put_attributes(*domain_or_name*, *items*, *replace=True*)
Store attributes for multiple items in a domain.

Parameters

- **domain_or_name** (string or *boto.sdb.domain.Domain* object.) – Either the name of a domain or a Domain object
- **items** (dict *or dict-like object*) – A dictionary-like object. The keys of the dictionary are the item names and the values are themselves dictionaries of attribute names/values, exactly the same as the attribute_names parameter of the scalar put_attributes call.
- **replace** (*bool*) – Whether the attribute values passed in will replace existing values or will be added as addition values. Defaults to True.

Return type bool

Returns True if successful

create_domain(*domain_name*)
Create a SimpleDB domain.

Parameters **domain_name** (*string*) – The name of the new domain

Return type *boto.sdb.domain.Domain* object

Returns The newly created domain

delete_attributes(*domain_or_name*, *item_name*, *attr_names=None*, *expected_value=None*)
Delete attributes from a given item in a domain.

Parameters

- **domain_or_name** (string or *boto.sdb.domain.Domain* object.) – Either the name of a domain or a Domain object
- **item_name** (*string*) – The name of the item whose attributes are being deleted.
- **attributes** (dict, list or *boto.sdb.item.Item*) – Either a list containing attribute names which will cause all values associated with that attribute name to be deleted or a dict or Item containing the attribute names and keys and list of values to delete as the value. If no value is supplied, all attribute name/values for the item will be deleted.
- **expected_value** (list) – If supplied, this is a list or tuple consisting of a single attribute name and expected value. The list can be of the form:
  - ['name', 'value']

  In which case the call will first verify that the attribute "name" of this item has a value of "value". If it does, the delete will proceed, otherwise a ConditionalCheckFailed error will be returned. The list can also be of the form:
  - ['name', True|False]

  which will simply check for the existence (True) or non-existence (False) of the attribute.

Return type bool

Returns True if successful

delete_domain(*domain_or_name*)
Delete a SimpleDB domain.

Caution: This will delete the domain and all items within the domain.

Parameters **domain_or_name** (string or *boto.sdb.domain.Domain* object.) – Either the name of a domain or a Domain object

Return type bool

Returns True if successful

domain_metadata(*domain_or_name*)

Get the Metadata for a SimpleDB domain.

Parameters **domain_or_name** (string or *boto.sdb.domain.Domain* object.) – Either the name of a domain or a Domain object

Return type *boto.sdb.domain.DomainMetaData* object

Returns The newly created domain metadata object

get_all_domains(*max_domains=None*, *next_token=None*)

Returns a *boto.resultset.ResultSet* containing all *boto.sdb.domain.Domain* objects associated with this connection's Access Key ID.

Parameters

- **max_domains** (*int*) – Limit the returned *ResultSet* to the specified number of members.
- **next_token** (*str*) – A token string that was returned in an earlier call to this method as the `next_token` attribute on the returned *ResultSet* object. This attribute is set if there are more than Domains than the value specified in the `max_domains` keyword. Pass the `next_token` value from you earlier query in this keyword to get the next 'page' of domains.

get_attributes(*domain_or_name*, *item_name*, *attribute_names=None*, *consistent_read=False*, *item=None*)

Retrieve attributes for a given item in a domain.

Parameters

- **domain_or_name** (string or *boto.sdb.domain.Domain* object.) – Either the name of a domain or a Domain object
- **item_name** (*string*) – The name of the item whose attributes are being retrieved.
- **attribute_names** (*string or list of strings*) – An attribute name or list of attribute names. This parameter is optional. If not supplied, all attributes will be retrieved for the item.
- **consistent_read** (*bool*) – When set to true, ensures that the most recent data is returned.
- **item** (*boto.sdb.item.Item*) – Instead of instantiating a new Item object, you may specify one to update.

Return type *boto.sdb.item.Item*

Returns An Item with the requested attribute name/values set on it

get_domain(*domain_name*, *validate=True*)

Retrieves a *boto.sdb.domain.Domain* object whose name matches `domain_name`.

Parameters

- **domain_name** (*str*) – The name of the domain to retrieve
- **validate** (*bool*) – When `True`, check to see if the domain actually exists. If `False`, blindly return a *Domain* object with the specified name set.

Raises *boto.exception.SDBResponseError* if `validate` is `True` and no match could be found.

Return type *boto.sdb.domain.Domain*

Returns The requested domain

get_domain_and_name(*domain_or_name*)

Given a `str` or *boto.sdb.domain.Domain*, return a `tuple` with the following members (in order):

•In instance of *boto.sdb.domain.Domain* for the requested domain

•The domain's name as a `str`

Parameters domain_or_name (`str` or *boto.sdb.domain.Domain*) – The domain or domain name to get the domain and name for.

Raises *boto.exception.SDBResponseError* when an invalid domain name is specified.

Return type tuple

Returns A `tuple` with contents outlined as per above.

get_usage()

Returns the BoxUsage (in USD) accumulated on this specific SDBConnection instance.

Tip: This can be out of date, and should only be treated as a rough estimate. Also note that this estimate only applies to the requests made on this specific connection instance. It is by no means an account-wide estimate.

Return type float

Returns The accumulated BoxUsage of all requests made on the connection.

lookup(*domain_name*, *validate=True*)

Lookup an existing SimpleDB domain. This differs from *get_domain()* in that `None` is returned if `validate` is `True` and no match was found (instead of raising an exception).

Parameters

- **domain_name** (*str*) – The name of the domain to retrieve
- **validate** (*bool*) – If `True`, a `None` value will be returned if the specified domain can't be found. If `False`, a *Domain* object will be dumbly returned, regardless of whether it actually exists.

Return type *boto.sdb.domain.Domain* object or `None`

Returns The Domain object or `None` if the domain does not exist.

print_usage()

Print the BoxUsage and approximate costs of all requests made on this specific SDBConnection instance.

Tip: This can be out of date, and should only be treated as a rough estimate. Also note that this estimate only applies to the requests made on this specific connection instance. It is by no means an account-wide estimate.

put_attributes(*domain_or_name*, *item_name*, *attributes*, *replace=True*, *expected_value=None*)
Store attributes for a given item in a domain.

Parameters

- **domain_or_name** (string or `boto.sdb.domain.Domain` object.) – Either the name of a domain or a Domain object
- **item_name** (*string*) – The name of the item whose attributes are being stored.
- **attribute_names** (`dict or dict-like object`) – The name/value pairs to store as attributes
- **expected_value** (`list`) – If supplied, this is a list or tuple consisting of a single attribute name and expected value. The list can be of the form:
  - ['name', 'value']

  In which case the call will first verify that the attribute "name" of this item has a value of "value". If it does, the delete will proceed, otherwise a ConditionalCheckFailed error will be returned. The list can also be of the form:
  - ['name', True|False]

  which will simply check for the existence (True) or non-existence (False) of the attribute.
- **replace** (*bool*) – Whether the attribute values passed in will replace existing values or will be added as addition values. Defaults to True.

Return type bool

Returns True if successful

select(*domain_or_name*, *query=''*, *next_token=None*, *consistent_read=False*)
Returns a set of Attributes for item names within domain_name that match the query. The query must be expressed in using the SELECT style syntax rather than the original SimpleDB query language. Even though the select request does not require a domain object, a domain object must be passed into this method so the Item objects returned can point to the appropriate domain.

Parameters

- **domain_or_name** (string or `boto.sdb.domain.Domain` object) – Either the name of a domain or a Domain object
- **query** (*string*) – The SimpleDB query to be performed.
- **consistent_read** (*bool*) – When set to true, ensures that the most recent data is returned.

Return type *ResultSet*

Returns An iterator containing the results.

set_item_cls(*cls*)
While the default item class is `boto.sdb.item.Item`, this default may be overridden. Use this method to change a connection's item class.

Parameters cls (*object*) – The new class to set as this connection's item class. See the default item class for inspiration as to what your replacement should/could look like.

boto.sdb.domain

Represents an SDB Domain

class `boto.sdb.domain.`**`Domain`**(*connection=None*, *name=None*)

`batch_delete_attributes`(*items*)

Delete multiple items in this domain.

Parameters **`items`** (`dict or dict-like object`) – A dictionary-like object. The keys of the dictionary are the item names and the values are either:

- dictionaries of attribute names/values, exactly the same as the attribute_names parameter of the scalar put_attributes call. The attribute name/value pairs will only be deleted if they match the name/value pairs passed in.
- None which means that all attributes associated with the item should be deleted.

Return type bool

Returns True if successful

`batch_put_attributes`(*items*, *replace=True*)

Store attributes for multiple items.

Parameters

- **`items`** (`dict or dict-like object`) – A dictionary-like object. The keys of the dictionary are the item names and the values are themselves dictionaries of attribute names/values, exactly the same as the attribute_names parameter of the scalar put_attributes call.
- **`replace`** (`bool`) – Whether the attribute values passed in will replace existing values or will be added as addition values. Defaults to True.

Return type bool

Returns True if successful

`delete`()

Delete this domain, and all items under it

`delete_attributes`(*item_name*, *attributes=None*, *expected_values=None*)

Delete attributes from a given item.

Parameters

- **`item_name`** (`string`) – The name of the item whose attributes are being deleted.
- **`attributes`** (dict, list or `boto.sdb.item.Item`) – Either a list containing attribute names which will cause all values associated with that attribute name to be deleted or a dict or Item containing the attribute names and keys and list of values to delete as the value. If no value is supplied, all attribute name/values for the item will be deleted.
- **`expected_value`** (`list`) – If supplied, this is a list or tuple consisting of a single attribute name and expected value. The list can be of the form:

  – [’name’, ‘value’]

  In which case the call will first verify that the attribute “name” of this item has a value of “value”. If it does, the delete will proceed, otherwise a ConditionalCheckFailed error will be returned. The list can also be of the form:

  – [’name’, True|False]

which will simply check for the existence (True) or non-existence (False) of the attribute.

Return type bool

Returns True if successful

`delete_item`(*item*)

`endElement`(*name*, *value*, *connection*)

`from_xml`(*doc*)
Load this domain based on an XML document

`get_attributes`(*item_name*, *attribute_name=None*, *consistent_read=False*, *item=None*)
Retrieve attributes for a given item.

Parameters

- **`item_name`** (*`string`*) – The name of the item whose attributes are being retrieved.
- **`attribute_names`** (*`string or list of strings`*) – An attribute name or list of attribute names. This parameter is optional. If not supplied, all attributes will be retrieved for the item.

Return type *`boto.sdb.item.Item`*

Returns An Item mapping type containing the requested attribute name/values

`get_item`(*item_name*, *consistent_read=False*)
Retrieves an item from the domain, along with all of its attributes.

Parameters

- **`item_name`** (*`string`*) – The name of the item to retrieve.
- **`consistent_read`** (*`bool`*) – When set to true, ensures that the most recent data is returned.

Return type *`boto.sdb.item.Item`* or `None`

Returns The requested item, or `None` if there was no match found

`get_metadata`()

`new_item`(*item_name*)

`put_attributes`(*item_name*, *attributes*, *replace=True*, *expected_value=None*)
Store attributes for a given item.

Parameters

- **`item_name`** (*`string`*) – The name of the item whose attributes are being stored.
- **`attribute_names`** (`dict` *`or dict-like object`*) – The name/value pairs to store as attributes
- **`expected_value`** (`list`) – If supplied, this is a list or tuple consisting of a single attribute name and expected value. The list can be of the form:
  - ['name', 'value']

  In which case the call will first verify that the attribute "name" of this item has a value of "value". If it does, the delete will proceed, otherwise a ConditionalCheckFailed error will be returned. The list can also be of the form:
  - ['name', True|False]

  which will simply check for the existence (True) or non-existence (False) of the attribute.

- **replace** (*bool*) – Whether the attribute values passed in will replace existing values or will be added as addition values. Defaults to True.

Return type bool

Returns True if successful

select (*query=''*, *next_token=None*, *consistent_read=False*, *max_items=None*)
Returns a set of Attributes for item names within domain_name that match the query. The query must be expressed in using the SELECT style syntax rather than the original SimpleDB query language.

Parameters **query** (*string*) – The SimpleDB query to be performed.

Return type iter

Returns An iterator containing the results. This is actually a generator function that will iterate across all search results, not just the first page.

startElement (*name*, *attrs*, *connection*)

to_xml (*f=None*)
Get this domain as an XML DOM Document :param f: Optional File to dump directly to :type f: File or Stream

Returns File object where the XML has been dumped to

Return type *file*

class boto.sdb.domain.**DomainDumpParser** (*domain*)
SAX parser for a domain that has been dumped

characters (*ch*)

endElement (*name*)

startElement (*name*, *attrs*)

class boto.sdb.domain.**DomainMetaData** (*domain=None*)

endElement (*name*, *value*, *connection*)

startElement (*name*, *attrs*, *connection*)

class boto.sdb.domain.**UploaderThread** (*domain*)
Uploader Thread

run ()

boto.sdb.item

class boto.sdb.item.**Item** (*domain*, *name=''*, *active=False*)
A `dict` sub-class that serves as an object representation of a SimpleDB item. An item in SDB is similar to a row in a relational database. Items belong to a `Domain`, which is similar to a table in a relational database.

The keys on instances of this object correspond to attributes that are stored on the SDB item.

Tip: While it is possible to instantiate this class directly, you may want to use the convenience methods on `boto.sdb.domain.Domain` for that purpose. For example, `boto.sdb.domain.Domain.get_item()`.

Parameters

- **domain** (*boto.sdb.domain.Domain*) – The domain that this item belongs to.
- **name** (*str*) – The name of this item. This name will be used when querying for items using methods like *boto.sdb.domain.Domain.get_item()*

add_value (*key*, *value*)
Helps set or add to attributes on this item. If you are adding a new attribute that has yet to be set, it will simply create an attribute named `key` with your given `value` as its value. If you are adding a value to an existing attribute, this method will convert the attribute to a list (if it isn't already) and append your new value to said list.

For clarification, consider the following interactive session:

```
>>> item = some_domain.get_item('some_item')
>>> item.has_key('some_attr')
False
>>> item.add_value('some_attr', 1)
>>> item['some_attr']
1
>>> item.add_value('some_attr', 2)
>>> item['some_attr']
[1, 2]
```

Parameters

- **key** (*str*) – The attribute to add a value to.
- **value** (*object*) – The value to set or append to the attribute.

decode_value (*value*)

delete ()
Deletes this item in SDB.

Note: This local Python object remains in its current state after deletion, this only deletes the remote item in SDB.

endElement (*name*, *value*, *connection*)

load ()
Loads or re-loads this item's attributes from SDB.

Warning: If you have changed attribute values on an Item instance, this method will over-write the values if they are different in SDB. For any local attributes that don't yet exist in SDB, they will be safe.

save (*replace=True*)
Saves this item to SDB.

Parameters **replace** (*bool*) – If `True`, delete any attributes on the remote SDB item that have a `None` value on this object.

startElement (*name*, *attrs*, *connection*)

boto.sdb.queryresultset

class `boto.sdb.queryresultset.`**`QueryResultSet`**(*domain=None*, *query=''*, *max_items=None*, *attr_names=None*)

class `boto.sdb.queryresultset.`**`SelectResultSet`**(*domain=None*, *query=''*, *max_items=None*, *next_token=None*, *consistent_read=False*)

> **`next`**()

`boto.sdb.queryresultset.`**`query_lister`**(*domain*, *query=''*, *max_items=None*, *attr_names=None*)

`boto.sdb.queryresultset.`**`select_lister`**(*domain*, *query=''*, *max_items=None*)

SDB DB Reference

This module offers an ORM-like layer on top of SimpleDB.

boto.sdb.db

boto.sdb.db.blob

class `boto.sdb.db.blob.`**`Blob`**(*value=None*, *file=None*, *id=None*)

> Blob object
>
> **`file`**
>
> **`next`**()
>
> **`read`**()
>
> **`readline`**()
>
> **`size`**

boto.sdb.db.key

class `boto.sdb.db.key.`**`Key`**(*encoded=None*, *obj=None*)

> **`app`**()
>
> **classmethod** **`from_path`**(**args*, ***kwds*)
>
> **`has_id_or_name`**()
>
> **`id`**()
>
> **`id_or_name`**()
>
> **`kind`**()
>
> **`name`**()
>
> **`parent`**()

boto.sdb.db.manager

boto.sdb.db.manager.**get_manager**(*cls*)

Returns the appropriate Manager class for a given Model class. It does this by looking in the boto config for a section like this:

```
[DB]
db_type = SimpleDB
db_user = <aws access key id>
db_passwd = <aws secret access key>
db_name = my_domain
[DB_TestBasic]
db_type = SimpleDB
db_user = <another aws access key id>
db_passwd = <another aws secret access key>
db_name = basic_domain
db_port = 1111
```

The values in the DB section are "generic values" that will be used if nothing more specific is found. You can also create a section for a specific Model class that gives the db info for that class. In the example above, TestBasic is a Model subclass.

boto.sdb.db.manager.pgmanager

Note: This module requires psycopg2 to be installed in the Python path.

boto.sdb.db.manager.sdbmanager

class boto.sdb.db.manager.sdbmanager.**SDBConverter**(*manager*)

Responsible for converting base Python types to format compatible with underlying database. For SimpleDB, that means everything needs to be converted to a string when stored in SimpleDB and from a string when retrieved.

To convert a value, pass it to the encode or decode method. The encode method will take a Python native value and convert to DB format. The decode method will take a DB format value and convert it to Python native format. To find the appropriate method to call, the generic encode/decode methods will look for the type-specific method by searching for a method called "encode_<type name>" or "decode_<type name>".

decode(*item_type*, *value*)

decode_blob(*value*)

decode_bool(*value*)

decode_date(*value*)

decode_datetime(*value*)

decode_float(*value*)

decode_int(*value*)

decode_list(*prop*, *value*)

decode_long(*value*)

decode_map(*prop*, *value*)

decode_map_element (*item_type*, *value*)
Decode a single element for a map

decode_prop (*prop*, *value*)

decode_reference (*value*)

decode_string (*value*)
Decoding a string is really nothing, just return the value as-is

decode_time (*value*)
converts strings in the form of HH:MM:SS.mmmmmm (created by datetime.time.isoformat()) to datetime.time objects.

Timzone-aware strings ("HH:MM:SS.mmmmmm+HH:MM") won't be handled right now and will raise TimeDecodeError.

encode (*item_type*, *value*)

encode_blob (*value*)

encode_bool (*value*)

encode_date (*value*)

encode_datetime (*value*)

encode_float (*value*)
See http://tools.ietf.org/html/draft-wood-ldapext-float-00.

encode_int (*value*)

encode_list (*prop*, *value*)

encode_long (*value*)

encode_map (*prop*, *value*)

encode_prop (*prop*, *value*)

encode_reference (*value*)

encode_string (*value*)
Convert ASCII, Latin-1 or UTF-8 to pure Unicode

encode_time (*value*)

class boto.sdb.db.manager.sdbmanager.**SDBManager** (*cls*, *db_name*, *db_user*, *db_passwd*, *db_host*, *db_port*, *db_table*, *ddl_dir*, *enable_ssl*, *consistent=None*)

count (*cls*, *filters*, *quick=True*, *sort_by=None*, *select=None*)
Get the number of results that would be returned in this query

decode_value (*prop*, *value*)

delete_key_value (*obj*, *name*)

delete_object (*obj*)

domain

encode_value (*prop*, *value*)

get_blob_bucket (*bucket_name=None*)

get_key_value (*obj*, *name*)

get_object (*cls*, *id*, *a=None*)

get_object_from_id (*id*)

get_property (*prop*, *obj*, *name*)

get_raw_item (*obj*)

get_s3_connection ()

load_object (*obj*)

query (*query*)

query_gql (*query_string*, **args*, ***kwds*)

save_object (*obj*, *expected_value=None*)

sdb

set_key_value (*obj*, *name*, *value*)

set_property (*prop*, *obj*, *name*, *value*)

exception boto.sdb.db.manager.sdbmanager.**TimeDecodeError**

boto.sdb.db.manager.xmlmanager

class boto.sdb.db.manager.xmlmanager.**XMLConverter** (*manager*)

Responsible for converting base Python types to format compatible with underlying database. For SimpleDB, that means everything needs to be converted to a string when stored in SimpleDB and from a string when retrieved.

To convert a value, pass it to the encode or decode method. The encode method will take a Python native value and convert to DB format. The decode method will take a DB format value and convert it to Python native format. To find the appropriate method to call, the generic encode/decode methods will look for the type-specific method by searching for a method called "encode_<type name>" or "decode_<type name>".

decode (*item_type*, *value*)

decode_bool (*value*)

decode_datetime (*value*)

decode_int (*value*)

decode_long (*value*)

decode_password (*value*)

decode_prop (*prop*, *value*)

decode_reference (*value*)

encode (*item_type*, *value*)

encode_bool (*value*)

encode_datetime (*value*)

encode_int (*value*)

encode_long (*value*)

encode_password (*value*)

encode_prop (*prop*, *value*)

encode_reference (*value*)

get_text_value (*parent_node*)

class boto.sdb.db.manager.xmlmanager.**XMLManager** (*cls*, *db_name*, *db_user*, *db_passwd*, *db_host*, *db_port*, *db_table*, *ddl_dir*, *enable_ssl*)

decode_value (*prop*, *value*)

delete_key_value (*obj*, *name*)

delete_object (*obj*)

encode_value (*prop*, *value*)

get_doc ()

get_key_value (*obj*, *name*)

get_list (*prop_node*, *item_type*)

get_object (*cls*, *id*)

get_object_from_doc (*cls*, *id*, *doc*)

get_property (*prop*, *obj*, *name*)

get_props_from_doc (*cls*, *id*, *doc*)
Pull out the properties from this document Returns the class, the properties in a hash, and the id if provided as a tuple :return: (cls, props, id)

get_raw_item (*obj*)

get_s3_connection ()

load_object (*obj*)

marshal_object (*obj*, *doc=None*)

new_doc ()

query (*cls*, *filters*, *limit=None*, *order_by=None*)

query_gql (*query_string*, **args*, ***kwds*)

reset ()

save_list (*doc*, *items*, *prop_node*)

save_object (*obj*, *expected_value=None*)
Marshal the object and do a PUT

set_key_value (*obj*, *name*, *value*)

set_property (*prop*, *obj*, *name*, *value*)

unmarshal_object (*fp*, *cls=None*, *id=None*)

unmarshal_props (*fp*, *cls=None*, *id=None*)
Same as unmarshalling an object, except it returns from "get_props_from_doc"

boto.sdb.db.model

class boto.sdb.db.model.**Expando** (*id=None*, ***kw*)

class `boto.sdb.db.model.`**`Model`**(*id=None*, ***kw*)

classmethod **`all`**(*limit=None*, *next_token=None*)

`delete`()

`delete_attributes`(*attrs*)

Delete just these attributes, not the whole object.

Parameters **`attrs`** (`list`) – Attributes to save, as a list of string names

Returns self

Return type *`boto.sdb.db.model.Model`*

classmethod **`find`**(*limit=None*, *next_token=None*, ***params*)

classmethod **`find_property`**(*prop_name*)

classmethod **`find_subclass`**(*name*)

Find a subclass with a given name

classmethod **`from_xml`**(*fp*)

classmethod **`get_by_id`**(*ids=None*, *parent=None*)

classmethod **`get_by_ids`**(*ids=None*, *parent=None*)

classmethod **`get_by_key_name`**(*key_names*, *parent=None*)

classmethod **`get_lineage`**()

classmethod **`get_or_insert`**(*key_name*, ***kw*)

classmethod **`get_xmlmanager`**()

`id` **= None**

`key`()

classmethod **`kind`**()

`load`()

classmethod **`properties`**(*hidden=True*)

`put`(*expected_value=None*)

Save this object as it is, with an optional expected value

Parameters **`expected_value`** (*`tuple or list`*) – Optional tuple of Attribute, and Value that must be the same in order to save this object. If this condition is not met, an SDBResponseError will be raised with a Confict status code.

Returns This object

Return type *`boto.sdb.db.model.Model`*

`put_attributes`(*attrs*)

Save just these few attributes, not the whole object

Parameters **`attrs`** (`dict`) – Attributes to save, key->value dict

Returns self

Return type *`boto.sdb.db.model.Model`*

`reload`()

save (*expected_value=None*)

Save this object as it is, with an optional expected value

Parameters **expected_value** (`tuple or list`) – Optional tuple of Attribute, and Value that must be the same in order to save this object. If this condition is not met, an SDBResponseError will be raised with a Confict status code.

Returns This object

Return type `boto.sdb.db.model.Model`

save_attributes (*attrs*)

Save just these few attributes, not the whole object

Parameters **attrs** (`dict`) – Attributes to save, key->value dict

Returns self

Return type `boto.sdb.db.model.Model`

set_manager (*manager*)

to_dict ()

to_xml (*doc=None*)

class boto.sdb.db.model.**ModelMeta** (*name*, *bases*, *dict*)

Metaclass for all Models

boto.sdb.db.property

class boto.sdb.db.property.**BlobProperty** (*verbose_name=None*, *name=None*, *default=None*, *required=False*, *validator=None*, *choices=None*, *unique=False*)

data_type

alias of `Blob`

type_name = 'blob'

class boto.sdb.db.property.**BooleanProperty** (*verbose_name=None*, *name=None*, *default=False*, *required=False*, *validator=None*, *choices=None*, *unique=False*)

data_type

alias of `bool`

empty (*value*)

type_name = 'Boolean'

class boto.sdb.db.property.**CalculatedProperty** (*verbose_name=None*, *name=None*, *default=None*, *required=False*, *validator=None*, *choices=None*, *calculated_type=<type 'int'>*, *unique=False*, *use_method=False*)

get_value_for_datastore (*model_instance*)

class `boto.sdb.db.property.`**`DateProperty`**(*verbose_name=None*, *auto_now=False*, *auto_now_add=False*, *name=None*, *default=None*, *required=False*, *validator=None*, *choices=None*, *unique=False*)

`data_type`
alias of `date`

`default_value`()

`get_value_for_datastore`(*model_instance*)

`now`()

`type_name` = **'Date'**

`validate`(*value*)

class `boto.sdb.db.property.`**`DateTimeProperty`**(*verbose_name=None*, *auto_now=False*, *auto_now_add=False*, *name=None*, *default=None*, *required=False*, *validator=None*, *choices=None*, *unique=False*)

`data_type`
alias of `datetime`

`default_value`()

`get_value_for_datastore`(*model_instance*)

`now`()

`type_name` = **'DateTime'**

`validate`(*value*)

class `boto.sdb.db.property.`**`FloatProperty`**(*verbose_name=None*, *name=None*, *default=0.0*, *required=False*, *validator=None*, *choices=None*, *unique=False*)

`data_type`
alias of `float`

`empty`(*value*)

`type_name` = **'Float'**

`validate`(*value*)

class `boto.sdb.db.property.`**`IntegerProperty`**(*verbose_name=None*, *name=None*, *default=0*, *required=False*, *validator=None*, *choices=None*, *unique=False*, *max=2147483647*, *min=-2147483648*)

`data_type`
alias of `int`

`empty`(*value*)

`type_name` = **'Integer'**

`validate`(*value*)

class `boto.sdb.db.property.`**`ListProperty`**(*item_type*, *verbose_name=None*, *name=None*, *default=None*, ***kwds*)

data_type
alias of `list`

default_value()

empty(*value*)

type_name = 'List'

validate(*value*)

class `boto.sdb.db.property.`**LongProperty**(*verbose_name=None*, *name=None*, *default=0*, *required=False*, *validator=None*, *choices=None*, *unique=False*)

data_type
alias of `long`

empty(*value*)

type_name = 'Long'

validate(*value*)

class `boto.sdb.db.property.`**MapProperty**(*item_type=<type 'str'>*, *verbose_name=None*, *name=None*, *default=None*, ***kwds*)

data_type
alias of `dict`

default_value()

empty(*value*)

type_name = 'Map'

validate(*value*)

class `boto.sdb.db.property.`**PasswordProperty**(*verbose_name=None*, *name=None*, *default=''*, *required=False*, *validator=None*, *choices=None*, *unique=False*, *hashfunc=None*)

Hashed property whose original value can not be retrieved, but still can be compared.

Works by storing a hash of the original value instead of the original value. Once that's done all that can be retrieved is the hash.

The comparison

obj.password == 'foo'

generates a hash of 'foo' and compares it to the stored hash.

Underlying data type for hashing, storing, and comparing is boto.utils.Password. The default hash function is defined there (currently sha512 in most cases, md5 where sha512 is not available)

It's unlikely you'll ever need to use a different hash function, but if you do, you can control the behavior in one of two ways:

1.Specifying hashfunc in PasswordProperty constructor

import hashlib

class MyModel(model): password = PasswordProperty(hashfunc=hashlib.sha224)

2.Subclassing Password and PasswordProperty

class SHA224Password(Password): hashfunc=hashlib.sha224

class SHA224PasswordProperty(PasswordProperty): data_type=MyPassword
type_name="MyPassword"

class MyModel(Model): password = SHA224PasswordProperty()

The hashfunc parameter overrides the default hashfunc in boto.utils.Password.

The remaining parameters are passed through to StringProperty.__init__

`data_type`
alias of `Password`

`get_value_for_datastore`(*model_instance*)

`make_value_from_datastore`(*value*)

`type_name` = 'Password'

`validate`(*value*)

class `boto.sdb.db.property.`**`Property`**(*verbose_name=None*, *name=None*, *default=None*, *required=False*, *validator=None*, *choices=None*, *unique=False*)

`data_type`
alias of `str`

`default_validator`(*value*)

`default_value`()

`empty`(*value*)

`get_choices`()

`get_value_for_datastore`(*model_instance*)

`make_value_from_datastore`(*value*)

`name` = ''

`type_name` = ''

`validate`(*value*)

`verbose_name` = ''

class `boto.sdb.db.property.`**`ReferenceProperty`**(*reference_class=None*, *collection_name=None*, *verbose_name=None*, *name=None*, *default=None*, *required=False*, *validator=None*, *choices=None*, *unique=False*)

`check_instance`(*value*)

`check_uuid`(*value*)

`data_type`
alias of `Key`

`type_name` = 'Reference'

`validate`(*value*)

class `boto.sdb.db.property.`**`S3KeyProperty`**(*verbose_name=None*, *name=None*, *default=None*, *required=False*, *validator=None*, *choices=None*, *unique=False*)

data_type
alias of `Key`

get_value_for_datastore(*model_instance*)

type_name = 'S3Key'

validate(*value*)

validate_regex = '^s3:\/\/([^\/]*)\/(.*)$'

class `boto.sdb.db.property.`**StringProperty**(*verbose_name=None*, *name=None*, *default=''*, *required=False*, *validator=<function validate_string>*, *choices=None*, *unique=False*)

type_name = 'String'

class `boto.sdb.db.property.`**TextProperty**(*verbose_name=None*, *name=None*, *default=''*, *required=False*, *validator=None*, *choices=None*, *unique=False*, *max_length=None*)

type_name = 'Text'

validate(*value*)

class `boto.sdb.db.property.`**TimeProperty**(*verbose_name=None*, *name=None*, *default=None*, *required=False*, *validator=None*, *choices=None*, *unique=False*)

data_type
alias of `time`

type_name = 'Time'

validate(*value*)

`boto.sdb.db.property.`**validate_string**(*value*)

boto.sdb.db.query

class `boto.sdb.db.query.`**Query**(*model_class*, *limit=None*, *next_token=None*, *manager=None*)

count(*quick=True*)

fetch(*limit*, *offset=0*)
Not currently fully supported, but we can use this to allow them to set a limit in a chainable method

filter(*property_operator*, *value*)

get_next_token()

get_query()

next()

next_token

order(*key*)

set_next_token(*token*)

to_xml(*doc=None*)

An Introduction to boto's DynamoDB interface

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

Creating a Connection

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

```
>>> import boto
>>> conn = boto.connect_dynamodb(
        aws_access_key_id='<YOUR_AWS_KEY_ID>',
        aws_secret_access_key='<YOUR_AWS_SECRET_KEY>')
>>> conn
<boto.dynamodb.layer2.Layer2 object at 0x3fb3090>
```

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. More details on configuration can be found in *Boto Config*.

Note: At this time, Amazon DynamoDB is available only in the US-EAST-1 region. The `connect_dynamodb` method automatically connect to that region.

The *`boto.connect_dynamodb()`* functions returns a *`boto.dynamodb.layer2.Layer2`* instance, which is a high-level API for working with DynamoDB. Layer2 is a set of abstractions that sit atop the lower level *`boto.dynamodb.layer1.Layer1`* API, which closely mirrors the Amazon DynamoDB API. For the purpose of this tutorial, we'll just be covering Layer2.

Listing Tables

Now that we have a DynamoDB connection object, we can then query for a list of existing tables in that region:

```
>>> conn.list_tables()
['test-table', 'another-table']
```

Creating Tables

DynamoDB tables are created with the *`Layer2.create_table`* method. While DynamoDB's items (a rough equivalent to a relational DB's row) don't have a fixed schema, you do need to create a schema for the table's hash key element, and the optional range key element. This is explained in greater detail in DynamoDB's Data Model documentation.

We'll start by defining a schema that has a hash key and a range key that are both keys:

```
>>> message_table_schema = conn.create_schema(
        hash_key_name='forum_name',
        hash_key_proto_value='S',
        range_key_name='subject',
        range_key_proto_value='S'
    )
```

The next few things to determine are table name and read/write throughput. We'll defer explaining throughput to the DynamoDB's Provisioned Throughput docs.

We're now ready to create the table:

```
>>> table = conn.create_table(
        name='messages',
        schema=message_table_schema,
        read_units=10,
        write_units=10
    )
>>> table
Table(messages)
```

This returns a `boto.dynamodb.table.Table` instance, which provides simple ways to create (put), update, and delete items.

Getting a Table

To retrieve an existing table, use `Layer2.get_table`:

```
>>> conn.list_tables()
['test-table', 'another-table', 'messages']
>>> table = conn.get_table('messages')
>>> table
Table(messages)
```

`Layer2.get_table`, like `Layer2.create_table`, returns a `boto.dynamodb.table.Table` instance.

Describing Tables

To get a complete description of a table, use `Layer2.describe_table`:

```
>>> conn.list_tables()
['test-table', 'another-table', 'messages']
>>> conn.describe_table('messages')
{
    'Table': {
        'CreationDateTime': 1327117581.624,
        'ItemCount': 0,
        'KeySchema': {
            'HashKeyElement': {
                'AttributeName': 'forum_name',
                'AttributeType': 'S'
            },
            'RangeKeyElement': {
                'AttributeName': 'subject',
                'AttributeType': 'S'
            }
        },
        'ProvisionedThroughput': {
            'ReadCapacityUnits': 10,
            'WriteCapacityUnits': 10
        },
        'TableName': 'messages',
        'TableSizeBytes': 0,
        'TableStatus': 'ACTIVE'
    }
}
```

Adding Items

Continuing on with our previously created `messages` table, adding an:

```
>>> table = conn.get_table('messages')
>>> item_data = {
        'Body': 'http://url_to_lolcat.gif',
        'SentBy': 'User A',
        'ReceivedTime': '12/9/2011 11:36:03 PM',
    }
>>> item = table.new_item(
        # Our hash key is 'forum'
        hash_key='LOLCat Forum',
        # Our range key is 'subject'
        range_key='Check this out!',
        # This has the
        attrs=item_data
    )
```

The *`Table.new_item`* method creates a new *`boto.dynamodb.item.Item`* instance with your specified hash key, range key, and attributes already set. *`Item`* is a `dict` sub-class, meaning you can edit your data as such:

```
item['a_new_key'] = 'testing'
del item['a_new_key']
```

After you are happy with the contents of the item, use *`Item.put`* to commit it to DynamoDB:

```
>>> item.put()
```

Retrieving Items

Now, let's check if it got added correctly. Since DynamoDB works under an 'eventual consistency' mode, we need to specify that we wish a consistent read, as follows:

```
>>> table = conn.get_table('messages')
>>> item = table.get_item(
        # Your hash key was 'forum_name'
        hash_key='LOLCat Forum',
        # Your range key was 'subject'
        range_key='Check this out!'
    )
>>> item
{
    # Note that this was your hash key attribute (forum_name)
    'forum_name': 'LOLCat Forum',
    # This is your range key attribute (subject)
    'subject': 'Check this out!'
    'Body': 'http://url_to_lolcat.gif',
    'ReceivedTime': '12/9/2011 11:36:03 PM',
    'SentBy': 'User A',
}
```

Updating Items

To update an item's attributes, simply retrieve it, modify the value, then `Item.put` it again:

```
>>> table = conn.get_table('messages')
>>> item = table.get_item(
        hash_key='LOLCat Forum',
        range_key='Check this out!'
    )
>>> item['SentBy'] = 'User B'
>>> item.put()
```

Deleting Items

To delete items, use the `Item.delete` method:

```
>>> table = conn.get_table('messages')
>>> item = table.get_item(
        hash_key='LOLCat Forum',
        range_key='Check this out!'
    )
>>> item.delete()
```

Deleting Tables

There are two easy ways to delete a table. Through your top-level `Layer2` object:

```
>>> conn.delete_table(table)
```

Or by getting the table, then using `Table.delete`:

```
>>> table = conn.get_table('messages')
>>> table.delete()
```

DynamoDB

boto.dynamodb

boto.dynamodb.layer1

class `boto.dynamodb.layer1.`**`Layer1`**(*aws_access_key_id=None*, *aws_secret_access_key=None*, *is_secure=True*, *port=None*, *proxy=None*, *proxy_port=None*, *host=None*, *debug=0*, *session_token=None*)

This is the lowest-level interface to DynamoDB. Methods at this layer map directly to API requests and parameters to the methods are either simple, scalar values or they are the Python equivalent of the JSON input as defined in the DynamoDB Developer's Guide. All responses are direct decoding of the JSON response bodies to Python data structures via the json or simplejson modules.

Variables **`throughput_exceeded_events`** – An integer variable that keeps a running total of the number of ThroughputExceeded responses this connection has received from Amazon DynamoDB.

`DefaultHost` **= 'dynamodb.us-east-1.amazonaws.com'**
The default DynamoDB API endpoint to connect to.

`ResponseError`
alias of `DynamoDBResponseError`

`ServiceName` **= 'DynamoDB'**
The name of the Service

`SessionExpiredError` **= 'com.amazon.coral.service#ExpiredTokenException'**
The error response returned when session token has expired

`ThruputError` **= 'ProvisionedThroughputExceededException'**
The error response returned when provisioned throughput is exceeded

`Version` **= '20111205'**
DynamoDB API version.

`batch_get_item`(*request_items*, *object_hook=None*)
Return a set of attributes for a multiple items in multiple tables using their primary keys.

Parameters **`request_items`** (`dict`) – A Python version of the RequestItems data structure defined by DynamoDB.

`create_table`(*table_name*, *schema*, *provisioned_throughput*)
Add a new table to your account. The table name must be unique among those associated with the account issuing the request. This request triggers an asynchronous workflow to begin creating the table. When the workflow is complete, the state of the table will be ACTIVE.

Parameters

- **`table_name`** (`str`) – The name of the table to create.
- **`schema`** (`dict`) – A Python version of the KeySchema data structure as defined by DynamoDB
- **`provisioned_throughput`** (`dict`) – A Python version of the ProvisionedThroughput data structure defined by DynamoDB.

`delete_item`(*table_name*, *key*, *expected=None*, *return_values=None*, *object_hook=None*)
Delete an item and all of it's attributes by primary key. You can perform a conditional delete by specifying an expected rule.

Parameters

- **`table_name`** (`str`) – The name of the table containing the item.
- **`key`** (`dict`) – A Python version of the Key data structure defined by DynamoDB.
- **`expected`** (`dict`) – A Python version of the Expected data structure defined by DynamoDB.
- **`return_values`** (`str`) – Controls the return of attribute name-value pairs before then were changed. Possible values are: None or 'ALL_OLD'. If 'ALL_OLD' is specified and the item is overwritten, the content of the old item is returned.

`delete_table`(*table_name*)
Deletes the table and all of it's data. After this request the table will be in the DELETING state until DynamoDB completes the delete operation.

Parameters **`table_name`** (*`str`*) – The name of the table to delete.

`describe_table`(*table_name*)

Returns information about the table including current state of the table, primary key schema and when the table was created.

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

`get_item`(*table_name*, *key*, *attributes_to_get=None*, *consistent_read=False*, *object_hook=None*)

Return a set of attributes for an item that matches the supplied key.

Parameters

- **`table_name`** (*`str`*) – The name of the table containing the item.
- **`key`** (`dict`) – A Python version of the Key data structure defined by DynamoDB.
- **`attributes_to_get`** (`list`) – A list of attribute names. If supplied, only the specified attribute names will be returned. Otherwise, all attributes will be returned.
- **`consistent_read`** (*`bool`*) – If True, a consistent read request is issued. Otherwise, an eventually consistent request is issued.

`list_tables`(*limit=None*, *start_table=None*)

Return a list of table names associated with the current account and endpoint.

Parameters

- **`limit`** (*`int`*) – The maximum number of tables to return.
- **`limit`** – The name of the table that starts the list. If you ran a previous list_tables and not all results were returned, the response dict would include a LastEvaluatedTableName attribute. Use that value here to continue the listing.

`make_request`(*action*, *body=''*, *object_hook=None*)

Raises `DynamoDBExpiredTokenError` if the security token expires.

`put_item`(*table_name*, *item*, *expected=None*, *return_values=None*, *object_hook=None*)

Create a new item or replace an old item with a new item (including all attributes). If an item already exists in the specified table with the same primary key, the new item will completely replace the old item. You can perform a conditional put by specifying an expected rule.

Parameters

- **`table_name`** (*`str`*) – The name of the table in which to put the item.
- **`item`** (`dict`) – A Python version of the Item data structure defined by DynamoDB.
- **`expected`** (`dict`) – A Python version of the Expected data structure defined by DynamoDB.
- **`return_values`** (*`str`*) – Controls the return of attribute name-value pairs before then were changed. Possible values are: None or 'ALL_OLD'. If 'ALL_OLD' is specified and the item is overwritten, the content of the old item is returned.

`query`(*table_name*, *hash_key_value*, *range_key_conditions=None*, *attributes_to_get=None*, *limit=None*, *consistent_read=False*, *scan_index_forward=True*, *exclusive_start_key=None*, *object_hook=None*)

Perform a query of DynamoDB. This version is currently punting and expecting you to provide a full and correct JSON body which is passed as is to DynamoDB.

Parameters

- **`table_name`** (*`str`*) – The name of the table to query.

- **key** – A DynamoDB-style HashKeyValue.
- **range_key_conditions** (dict) – A Python version of the RangeKeyConditions data structure.
- **attributes_to_get** (list) – A list of attribute names. If supplied, only the specified attribute names will be returned. Otherwise, all attributes will be returned.
- **limit** (*int*) – The maximum number of items to return.
- **consistent_read** (*bool*) – If True, a consistent read request is issued. Otherwise, an eventually consistent request is issued.
- **scan_index_forward** (*bool*) – Specified forward or backward traversal of the index. Default is forward (True).
- **exclusive_start_key** (list *or* *tuple*) – Primary key of the item from which to continue an earlier query. This would be provided as the LastEvaluatedKey in that query.

scan(*table_name*, *scan_filter=None*, *attributes_to_get=None*, *limit=None*, *count=False*, *exclusive_start_key=None*, *object_hook=None*)

Perform a scan of DynamoDB. This version is currently punting and expecting you to provide a full and correct JSON body which is passed as is to DynamoDB.

Parameters

- **table_name** (*str*) – The name of the table to scan.
- **scan_filter** (dict) – A Python version of the ScanFilter data structure.
- **attributes_to_get** (list) – A list of attribute names. If supplied, only the specified attribute names will be returned. Otherwise, all attributes will be returned.
- **limit** (*int*) – The maximum number of items to return.
- **count** (*bool*) – If True, Amazon DynamoDB returns a total number of items for the Scan operation, even if the operation has no matching items for the assigned filter.
- **exclusive_start_key** (list *or* *tuple*) – Primary key of the item from which to continue an earlier query. This would be provided as the LastEvaluatedKey in that query.

update_item(*table_name*, *key*, *attribute_updates*, *expected=None*, *return_values=None*, *object_hook=None*)

Edits an existing item's attributes. You can perform a conditional update (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.
- **key** (dict) – A Python version of the Key data structure defined by DynamoDB which identifies the item to be updated.
- **attribute_updates** (dict) – A Python version of the AttributeUpdates data structure defined by DynamoDB.
- **expected** (dict) – A Python version of the Expected data structure defined by DynamoDB.
- **return_values** (*str*) – Controls the return of attribute name-value pairs before then were changed. Possible values are: None or 'ALL_OLD'. If 'ALL_OLD' is specified and the item is overwritten, the content of the old item is returned.

update_table (*table_name*, *provisioned_throughput*)
Updates the provisioned throughput for a given table.

Parameters

- **table_name** (*str*) – The name of the table to update.
- **provisioned_throughput** (*dict*) – A Python version of the ProvisionedThroughput data structure defined by DynamoDB.

boto.dynamodb.layer2

class boto.dynamodb.layer2.**Layer2** (*aws_access_key_id=None*, *aws_secret_access_key=None*, *is_secure=True*, *port=None*, *proxy=None*, *proxy_port=None*, *host=None*, *debug=0*, *session_token=None*)

batch_get_item (*batch_list*)
Return a set of attributes for a multiple items in multiple tables using their primary keys.

Parameters batch_list (*boto.dynamodb.batch.BatchList*) – A BatchList object which consists of a list of `boto.dynamoddb.batch.Batch` objects. Each Batch object contains the information about one batch of objects that you wish to retrieve in this request.

build_key_from_values (*schema*, *hash_key*, *range_key=None*)
Build a Key structure to be used for accessing items in Amazon DynamoDB. This method takes the supplied hash_key and optional range_key and validates them against the schema. If there is a mismatch, a TypeError is raised. Otherwise, a Python dict version of a Amazon DynamoDB Key data structure is returned.

Parameters

- **hash_key** (*int, float, str, or unicode*) – The hash key of the item you are looking for. The type of the hash key should match the type defined in the schema.
- **range_key** (*int, float, str or unicode*) – The range key of the item your are looking for. This should be supplied only if the schema requires a range key. The type of the range key should match the type defined in the schema.

create_schema (*hash_key_name*, *hash_key_proto_value*, *range_key_name=None*, *range_key_proto_value=None*)
Create a Schema object used when creating a Table.

Parameters

- **hash_key_name** (*str*) – The name of the HashKey for the schema.
- **hash_key_proto_value** (*int|long|float|str|unicode*) – A sample or prototype of the type of value you want to use for the HashKey.
- **range_key_name** (*str*) – The name of the RangeKey for the schema. This parameter is optional.
- **range_key_proto_value** (*int|long|float|str|unicode*) – A sample or prototype of the type of value you want to use for the RangeKey. This parameter is optional.

create_table (*name*, *schema*, *read_units*, *write_units*)
Create a new Amazon DynamoDB table.

Parameters

- **name** (*str*) – The name of the desired table.

- **schema** (*boto.dynamodb.schema.Schema*) – The Schema object that defines the schema used by this table.
- **read_units** (*int*) – The value for ReadCapacityUnits.
- **write_units** (*int*) – The value for WriteCapacityUnits.

Return type *boto.dynamodb.table.Table*

Returns A Table object representing the new Amazon DynamoDB table.

delete_item(*item*, *expected_value=None*, *return_values=None*)
Delete the item from Amazon DynamoDB.

Parameters

- **item** (*boto.dynamodb.item.Item*) – The Item to delete from Amazon DynamoDB.
- **expected_value** (*dict*) – A dictionary of name/value pairs that you expect. This dictionary should have name/value pairs where the name is the name of the attribute and the value is either the value you are expecting or False if you expect the attribute not to exist.
- **return_values** (*str*) – Controls the return of attribute name-value pairs before then were changed. Possible values are: None or 'ALL_OLD'. If 'ALL_OLD' is specified and the item is overwritten, the content of the old item is returned.

delete_table(*table*)
Delete this table and all items in it. After calling this the Table objects status attribute will be set to 'DELETING'.

Parameters table (*boto.dynamodb.table.Table*) – The Table object that is being deleted.

describe_table(*name*)
Retrieve information about an existing table.

Parameters name (*str*) – The name of the desired table.

dynamize_attribute_updates(*pending_updates*)
Convert a set of pending item updates into the structure required by Layer1.

dynamize_expected_value(*expected_value*)
Convert an expected_value parameter into the data structure required for Layer1.

dynamize_item(*item*)

dynamize_last_evaluated_key(*last_evaluated_key*)
Convert a last_evaluated_key parameter into the data structure required for Layer1.

dynamize_range_key_condition(*range_key_condition*)
Convert a layer2 range_key_condition parameter into the structure required by Layer1.

dynamize_request_items(*batch_list*)
Convert a request_items parameter into the data structure required for Layer1.

dynamize_scan_filter(*scan_filter*)
Convert a layer2 scan_filter parameter into the structure required by Layer1.

dynamize_value(*val*)
Take a scalar Python value and return a dict consisting of the Amazon DynamoDB type specification and the value that needs to be sent to Amazon DynamoDB. If the type of the value is not supported, raise a TypeError

get_dynamodb_type(*val*)
Take a scalar Python value and return a string representing the corresponding Amazon DynamoDB type. If the value passed in is not a supported type, raise a TypeError.

get_item(*table*, *hash_key*, *range_key=None*, *attributes_to_get=None*, *consistent_read=False*, *item_class=<class 'boto.dynamodb.item.Item'>*)
Retrieve an existing item from the table.

Parameters

- **table** (*boto.dynamodb.table.Table*) – The Table object from which the item is retrieved.
- **hash_key** (*int|long|float|str|unicode*) – The HashKey of the requested item. The type of the value must match the type defined in the schema for the table.
- **range_key** (*int|long|float|str|unicode*) – The optional RangeKey of the requested item. The type of the value must match the type defined in the schema for the table.
- **attributes_to_get** (*list*) – A list of attribute names. If supplied, only the specified attribute names will be returned. Otherwise, all attributes will be returned.
- **consistent_read** (*bool*) – If True, a consistent read request is issued. Otherwise, an eventually consistent request is issued.
- **item_class** (*Class*) – Allows you to override the class used to generate the items. This should be a subclass of *boto.dynamodb.item.Item*

get_table(*name*)
Retrieve the Table object for an existing table.

Parameters **name** (*str*) – The name of the desired table.

Return type *boto.dynamodb.table.Table*

Returns A Table object representing the table.

list_tables(*limit=None*, *start_table=None*)
Return a list of the names of all Tables associated with the current account and region. TODO - Layer2 should probably automatically handle pagination.

Parameters

- **limit** (*int*) – The maximum number of tables to return.
- **limit** – The name of the table that starts the list. If you ran a previous list_tables and not all results were returned, the response dict would include a LastEvaluatedTableName attribute. Use that value here to continue the listing.

lookup(*name*)
Retrieve the Table object for an existing table.

Parameters **name** (*str*) – The name of the desired table.

Return type *boto.dynamodb.table.Table*

Returns A Table object representing the table.

new_batch_list()
Return a new, empty *boto.dynamodb.batch.BatchList* object.

put_item(*item*, *expected_value=None*, *return_values=None*)
Store a new item or completely replace an existing item in Amazon DynamoDB.

Parameters

- **item** (*boto.dynamodb.item.Item*) – The Item to write to Amazon DynamoDB.
- **expected_value** (*dict*) – A dictionary of name/value pairs that you expect. This dictionary should have name/value pairs where the name is the name of the attribute and the value is either the value you are expecting or False if you expect the attribute not to exist.
- **return_values** (*str*) – Controls the return of attribute name-value pairs before then were changed. Possible values are: None or 'ALL_OLD'. If 'ALL_OLD' is specified and the item is overwritten, the content of the old item is returned.

query(*table*, *hash_key*, *range_key_condition=None*, *attributes_to_get=None*, *request_limit=None*, *max_results=None*, *consistent_read=False*, *scan_index_forward=True*, *exclusive_start_key=None*, *item_class=<class 'boto.dynamodb.item.Item'>*)

Perform a query on the table.

Parameters

- **table** (*boto.dynamodb.table.Table*) – The Table object that is being queried.
- **hash_key** (*int|long|float|str|unicode*) – The HashKey of the requested item. The type of the value must match the type defined in the schema for the table.
- **range_key_condition** (*dict*) – A dict where the key is either a scalar value appropriate for the RangeKey in the schema of the database or a tuple of such values. The value associated with this key in the dict will be one of the following conditions:

  'EQ'|'LE'|'LT'|'GE'|'GT'|'BEGINS_WITH'|'BETWEEN'

  The only condition which expects or will accept a tuple of values is 'BETWEEN', otherwise a scalar value should be used as the key in the dict.
- **attributes_to_get** (*list*) – A list of attribute names. If supplied, only the specified attribute names will be returned. Otherwise, all attributes will be returned.
- **request_limit** (*int*) – The maximum number of items to retrieve from Amazon DynamoDB on each request. You may want to set a specific request_limit based on the provisioned throughput of your table. The default behavior is to retrieve as many results as possible per request.
- **max_results** (*int*) – The maximum number of results that will be retrieved from Amazon DynamoDB in total. For example, if you only wanted to see the first 100 results from the query, regardless of how many were actually available, you could set max_results to 100 and the generator returned from the query method will only yeild 100 results max.
- **consistent_read** (*bool*) – If True, a consistent read request is issued. Otherwise, an eventually consistent request is issued.
- **scan_index_forward** (*bool*) – Specified forward or backward traversal of the index. Default is forward (True).
- **exclusive_start_key** (*list or tuple*) – Primary key of the item from which to continue an earlier query. This would be provided as the LastEvaluatedKey in that query.
- **item_class** (*Class*) – Allows you to override the class used to generate the items. This should be a subclass of `boto.dynamodb.item.Item`

Return type generator

scan(*table*, *scan_filter=None*, *attributes_to_get=None*, *request_limit=None*, *max_results=None*, *count=False*, *exclusive_start_key=None*, *item_class=<class 'boto.dynamodb.item.Item'>*)

Perform a scan of DynamoDB.

Parameters

- **table** (*boto.dynamodb.table.Table*) – The Table object that is being scanned.
- **scan_filter** (*A list of tuples*) – A list of tuples where each tuple consists of an attribute name, a comparison operator, and either a scalar or tuple consisting of the values to compare the attribute to. Valid comparison operators are shown below along with the expected number of values that should be supplied.
  - EQ - equal (1)
  - NE - not equal (1)
  - LE - less than or equal (1)
  - LT - less than (1)
  - GE - greater than or equal (1)
  - GT - greater than (1)
  - NOT_NULL - attribute exists (0, use None)
  - NULL - attribute does not exist (0, use None)
  - CONTAINS - substring or value in list (1)
  - NOT_CONTAINS - absence of substring or value in list (1)
  - BEGINS_WITH - substring prefix (1)
  - IN - exact match in list (N)
  - BETWEEN - >= first value, <= second value (2)
- **attributes_to_get** (*list*) – A list of attribute names. If supplied, only the specified attribute names will be returned. Otherwise, all attributes will be returned.
- **request_limit** (*int*) – The maximum number of items to retrieve from Amazon DynamoDB on each request. You may want to set a specific request_limit based on the provisioned throughput of your table. The default behavior is to retrieve as many results as possible per request.
- **max_results** (*int*) – The maximum number of results that will be retrieved from Amazon DynamoDB in total. For example, if you only wanted to see the first 100 results from the query, regardless of how many were actually available, you could set max_results to 100 and the generator returned from the query method will only yeild 100 results max.
- **count** (*bool*) – If True, Amazon DynamoDB returns a total number of items for the Scan operation, even if the operation has no matching items for the assigned filter.
- **exclusive_start_key** (*list or tuple*) – Primary key of the item from which to continue an earlier query. This would be provided as the LastEvaluatedKey in that query.
- **item_class** (*Class*) – Allows you to override the class used to generate the items. This should be a subclass of *boto.dynamodb.item.Item*

Return type generator

update_item(*item*, *expected_value=None*, *return_values=None*)
Commit pending item updates to Amazon DynamoDB.

Parameters

- **item** (*boto.dynamodb.item.Item*) – The Item to update in Amazon DynamoDB. It is expected that you would have called the add_attribute, put_attribute and/or delete_attribute methods on this Item prior to calling this method. Those queued changes are what will be updated.
- **expected_value** (dict) – A dictionary of name/value pairs that you expect. This dictionary should have name/value pairs where the name is the name of the attribute and the value is either the value you are expecting or False if you expect the attribute not to exist.
- **return_values** (*str*) – Controls the return of attribute name/value pairs before they were updated. Possible values are: None, 'ALL_OLD', 'UPDATED_OLD', 'ALL_NEW' or 'UPDATED_NEW'. If 'ALL_OLD' is specified and the item is overwritten, the content of the old item is returned. If 'ALL_NEW' is specified, then all the attributes of the new version of the item are returned. If 'UPDATED_NEW' is specified, the new versions of only the updated attributes are returned.

update_throughput (*table*, *read_units*, *write_units*)
Update the ProvisionedThroughput for the Amazon DynamoDB Table.

Parameters

- **table** (*boto.dynamodb.table.Table*) – The Table object whose throughput is being updated.
- **read_units** (*int*) – The new value for ReadCapacityUnits.
- **write_units** (*int*) – The new value for WriteCapacityUnits.

boto.dynamodb.layer2.**convert_num** (*s*)

boto.dynamodb.layer2.**is_num** (*n*)

boto.dynamodb.layer2.**is_str** (*n*)

boto.dynamodb.layer2.**item_object_hook** (*dct*)
A custom object hook for use when decoding JSON item bodys. This hook will transform Amazon DynamoDB JSON responses to something that maps directly to native Python types.

boto.dynamodb.table

class boto.dynamodb.table.**Table** (*layer2*, *response=None*)
An Amazon DynamoDB table.

Variables

- ***name*** – The name of the table.
- ***create_time*** – The date and time that the table was created.
- ***status*** – The current status of the table. One of: 'ACTIVE', 'UPDATING', 'DELETING'.
- ***schema*** – A *boto.dynamodb.schema.Schema* object representing the schema defined for the table.
- ***item_count*** – The number of items in the table. This value is set only when the Table object is created or refreshed and may not reflect the actual count.
- ***size_bytes*** – Total size of the specified table, in bytes. Amazon DynamoDB updates this value approximately every six hours. Recent changes might not be reflected in this value.
- ***read_units*** – The ReadCapacityUnits of the tables Provisioned Throughput.

- ***write_units*** – The WriteCapacityUnits of the tables Provisioned Throughput.
- ***schema*** – The Schema object associated with the table.

create_time

delete()
Delete this table and all items in it. After calling this the Table objects status attribute will be set to 'DELETING'.

get_item(*hash_key*, *range_key=None*, *attributes_to_get=None*, *consistent_read=False*, *item_class=<class 'boto.dynamodb.item.Item'>*)
Retrieve an existing item from the table.

Parameters

- **hash_key** (*int|long|float|str|unicode*) – The HashKey of the requested item. The type of the value must match the type defined in the schema for the table.
- **range_key** (*int|long|float|str|unicode*) – The optional RangeKey of the requested item. The type of the value must match the type defined in the schema for the table.
- **attributes_to_get** (list) – A list of attribute names. If supplied, only the specified attribute names will be returned. Otherwise, all attributes will be returned.
- **consistent_read** (*bool*) – If True, a consistent read request is issued. Otherwise, an eventually consistent request is issued.
- **item_class** (*Class*) – Allows you to override the class used to generate the items. This should be a subclass of *boto.dynamodb.item.Item*

has_item(*hash_key*, *range_key=None*, *consistent_read=False*)
Checks the table to see if the Item with the specified hash_key exists. This may save a tiny bit of time/bandwidth over a straight *get_item()* if you have no intention to touch the data that is returned, since this method specifically tells Amazon not to return anything but the Item's key.

Parameters

- **hash_key** (*int|long|float|str|unicode*) – The HashKey of the requested item. The type of the value must match the type defined in the schema for the table.
- **range_key** (*int|long|float|str|unicode*) – The optional RangeKey of the requested item. The type of the value must match the type defined in the schema for the table.
- **consistent_read** (*bool*) – If True, a consistent read request is issued. Otherwise, an eventually consistent request is issued.

Return type bool

Returns True if the Item exists, False if not.

item_count

lookup(*hash_key*, *range_key=None*, *attributes_to_get=None*, *consistent_read=False*, *item_class=<class 'boto.dynamodb.item.Item'>*)
Retrieve an existing item from the table.

Parameters

- **hash_key** (*int|long|float|str|unicode*) – The HashKey of the requested item. The type of the value must match the type defined in the schema for the table.

- **range_key** (*int|long|float|str|unicode*) – The optional RangeKey of the requested item. The type of the value must match the type defined in the schema for the table.
- **attributes_to_get** (list) – A list of attribute names. If supplied, only the specified attribute names will be returned. Otherwise, all attributes will be returned.
- **consistent_read** (*bool*) – If True, a consistent read request is issued. Otherwise, an eventually consistent request is issued.
- **item_class** (*Class*) – Allows you to override the class used to generate the items. This should be a subclass of *boto.dynamodb.item.Item*

name

new_item(*hash_key*, *range_key=None*, *attrs=None*)
Return an new, unsaved Item which can later be PUT to Amazon DynamoDB.

query(*hash_key*, *range_key_condition=None*, *attributes_to_get=None*, *request_limit=None*, *max_results=None*, *consistent_read=False*, *scan_index_forward=True*, *exclusive_start_key=None*, *item_class=<class 'boto.dynamodb.item.Item'>*)
Perform a query on the table.

Parameters

- **hash_key** (*int|long|float|str|unicode*) – The HashKey of the requested item. The type of the value must match the type defined in the schema for the table.
- **range_key_condition** (dict) – A dict where the key is either a scalar value appropriate for the RangeKey in the schema of the database or a tuple of such values. The value associated with this key in the dict will be one of the following conditions:

  'EQ'|'LE'|'LT'|'GE'|'GT'|'BEGINS_WITH'|'BETWEEN'

  The only condition which expects or will accept a tuple of values is 'BETWEEN', otherwise a scalar value should be used as the key in the dict.
- **attributes_to_get** (list) – A list of attribute names. If supplied, only the specified attribute names will be returned. Otherwise, all attributes will be returned.
- **request_limit** (*int*) – The maximum number of items to retrieve from Amazon DynamoDB on each request. You may want to set a specific request_limit based on the provisioned throughput of your table. The default behavior is to retrieve as many results as possible per request.
- **max_results** (*int*) – The maximum number of results that will be retrieved from Amazon DynamoDB in total. For example, if you only wanted to see the first 100 results from the query, regardless of how many were actually available, you could set max_results to 100 and the generator returned from the query method will only yeild 100 results max.
- **consistent_read** (*bool*) – If True, a consistent read request is issued. Otherwise, an eventually consistent request is issued.
- **scan_index_forward** (*bool*) – Specified forward or backward traversal of the index. Default is forward (True).
- **exclusive_start_key** (list *or* tuple) – Primary key of the item from which to continue an earlier query. This would be provided as the LastEvaluatedKey in that query.
- **item_class** (*Class*) – Allows you to override the class used to generate the items. This should be a subclass of *boto.dynamodb.item.Item*

read_units

refresh (*wait_for_active=False*, *retry_seconds=5*)
Refresh all of the fields of the Table object by calling the underlying DescribeTable request.

Parameters

- **wait_for_active** (*bool*) – If True, this command will not return until the table status, as returned from Amazon DynamoDB, is 'ACTIVE'.
- **retry_seconds** (*int*) – If wait_for_active is True, this parameter controls the number of seconds of delay between calls to update_table in Amazon DynamoDB. Default is 5 seconds.

scan (*scan_filter=None*, *attributes_to_get=None*, *request_limit=None*, *max_results=None*, *count=False*, *exclusive_start_key=None*, *item_class=<class 'boto.dynamodb.item.Item'>*)
Scan through this table, this is a very long and expensive operation, and should be avoided if at all possible.

Parameters

- **scan_filter** (*A list of tuples*) – A list of tuples where each tuple consists of an attribute name, a comparison operator, and either a scalar or tuple consisting of the values to compare the attribute to. Valid comparison operators are shown below along with the expected number of values that should be supplied.
  - EQ - equal (1)
  - NE - not equal (1)
  - LE - less than or equal (1)
  - LT - less than (1)
  - GE - greater than or equal (1)
  - GT - greater than (1)
  - NOT_NULL - attribute exists (0, use None)
  - NULL - attribute does not exist (0, use None)
  - CONTAINS - substring or value in list (1)
  - NOT_CONTAINS - absence of substring or value in list (1)
  - BEGINS_WITH - substring prefix (1)
  - IN - exact match in list (N)
  - BETWEEN - >= first value, <= second value (2)
- **attributes_to_get** (*list*) – A list of attribute names. If supplied, only the specified attribute names will be returned. Otherwise, all attributes will be returned.
- **request_limit** (*int*) – The maximum number of items to retrieve from Amazon DynamoDB on each request. You may want to set a specific request_limit based on the provisioned throughput of your table. The default behavior is to retrieve as many results as possible per request.
- **max_results** (*int*) – The maximum number of results that will be retrieved from Amazon DynamoDB in total. For example, if you only wanted to see the first 100 results from the query, regardless of how many were actually available, you could set max_results to 100 and the generator returned from the query method will only yeild 100 results max.
- **count** (*bool*) – If True, Amazon DynamoDB returns a total number of items for the Scan operation, even if the operation has no matching items for the assigned filter.

- **exclusive_start_key** (list *or* tuple) – Primary key of the item from which to continue an earlier query. This would be provided as the LastEvaluatedKey in that query.
- **item_class** (*Class*) – Allows you to override the class used to generate the items. This should be a subclass of *boto.dynamodb.item.Item*

Return type generator

schema

size_bytes

status

update_from_response (*response*)
Update the state of the Table object based on the response data received from Amazon DynamoDB.

update_throughput (*read_units*, *write_units*)
Update the ProvisionedThroughput for the Amazon DynamoDB Table.

Parameters

- **read_units** (*int*) – The new value for ReadCapacityUnits.
- **write_units** (*int*) – The new value for WriteCapacityUnits.

write_units

boto.dynamodb.schema

class boto.dynamodb.schema.**Schema** (*schema_dict*)
Represents a DynamoDB schema.

Variables

- ***hash_key_name*** – The name of the hash key of the schema.
- ***hash_key_type*** – The DynamoDB type specification for the hash key of the schema.
- ***range_key_name*** – The name of the range key of the schema or None if no range key is defined.
- ***range_key_type*** – The DynamoDB type specification for the range key of the schema or None if no range key is defined.
- ***dict*** – The underlying Python dictionary that needs to be passed to Layer1 methods.

dict

hash_key_name

hash_key_type

range_key_name

range_key_type

boto.dynamodb.item

class boto.dynamodb.item.**Item** (*table*, *hash_key=None*, *range_key=None*, *attrs=None*)
An item in Amazon DynamoDB.

Variables

- ***hash_key*** – The HashKey of this item.
- ***range_key*** – The RangeKey of this item or None if no RangeKey is defined.
- ***hash_key_name*** – The name of the HashKey associated with this item.
- ***range_key_name*** – The name of the RangeKey associated with this item.
- ***table*** – The Table this item belongs to.

add_attribute (*attr_name*, *attr_value*)

Queue the addition of an attribute to an item in DynamoDB. This will eventually result in an UpdateItem request being issued with an update action of ADD when the save method is called.

Parameters

- **attr_name** (*str*) – Name of the attribute you want to alter.
- **attr_value** (*int | long | float | set*) – Value which is to be added to the attribute.

delete (*expected_value=None*, *return_values=None*)

Delete the item from DynamoDB.

Parameters

- **expected_value** (dict) – A dictionary of name/value pairs that you expect. This dictionary should have name/value pairs where the name is the name of the attribute and the value is either the value you are expecting or False if you expect the attribute not to exist.
- **return_values** (*str*) – Controls the return of attribute name-value pairs before then were changed. Possible values are: None or 'ALL_OLD'. If 'ALL_OLD' is specified and the item is overwritten, the content of the old item is returned.

delete_attribute (*attr_name*, *attr_value=None*)

Queue the deletion of an attribute from an item in DynamoDB. This call will result in a UpdateItem request being issued with update action of DELETE when the save method is called.

Parameters

- **attr_name** (*str*) – Name of the attribute you want to alter.
- **attr_value** (set) – A set of values to be removed from the attribute. This parameter is optional. If None, the whole attribute is removed from the item.

hash_key

hash_key_name

put (*expected_value=None*, *return_values=None*)

Store a new item or completely replace an existing item in Amazon DynamoDB.

Parameters

- **expected_value** (dict) – A dictionary of name/value pairs that you expect. This dictionary should have name/value pairs where the name is the name of the attribute and the value is either the value you are expecting or False if you expect the attribute not to exist.
- **return_values** (*str*) – Controls the return of attribute name-value pairs before then were changed. Possible values are: None or 'ALL_OLD'. If 'ALL_OLD' is specified and the item is overwritten, the content of the old item is returned.

put_attribute(*attr_name*, *attr_value*)
Queue the putting of an attribute to an item in DynamoDB. This call will result in an UpdateItem request being issued with the update action of PUT when the save method is called.

Parameters

- **attr_name** (*str*) – Name of the attribute you want to alter.
- **attr_value** (*int|long|float|str|set*) – New value of the attribute.

range_key

range_key_name

save(*expected_value=None*, *return_values=None*)
Commits pending updates to Amazon DynamoDB.

Parameters

- **expected_value** (dict) – A dictionary of name/value pairs that you expect. This dictionary should have name/value pairs where the name is the name of the attribute and the value is either the value you are expecting or False if you expect the attribute not to exist.
- **return_values** (*str*) – Controls the return of attribute name/value pairs before they were updated. Possible values are: None, 'ALL_OLD', 'UPDATED_OLD', 'ALL_NEW' or 'UPDATED_NEW'. If 'ALL_OLD' is specified and the item is overwritten, the content of the old item is returned. If 'ALL_NEW' is specified, then all the attributes of the new version of the item are returned. If 'UPDATED_NEW' is specified, the new versions of only the updated attributes are returned.

boto.dynamodb.batch

class boto.dynamodb.batch.**Batch**(*table*, *keys*, *attributes_to_get=None*)

Variables

- ***table*** – The Table object from which the item is retrieved.
- ***keys*** – A list of scalar or tuple values. Each element in the list represents one Item to retrieve. If the schema for the table has both a HashKey and a RangeKey, each element in the list should be a tuple consisting of (hash_key, range_key). If the schema for the table contains only a HashKey, each element in the list should be a scalar value of the appropriate type for the table schema.
- **attributes_to_get** – A list of attribute names. If supplied, only the specified attribute names will be returned. Otherwise, all attributes will be returned.

class boto.dynamodb.batch.**BatchList**(*layer2*)
A subclass of a list object that contains a collection of *boto.dynamodb.batch.Batch* objects.

add_batch(*table*, *keys*, *attributes_to_get=None*)
Add a Batch to this BatchList.

Parameters

- **table** (*boto.dynamodb.table.Table*) – The Table object in which the items are contained.
- **keys** (list) – A list of scalar or tuple values. Each element in the list represents one Item to retrieve. If the schema for the table has both a HashKey and a RangeKey, each element in the list should be a tuple consisting of (hash_key, range_key). If the schema for

the table contains only a HashKey, each element in the list should be a scalar value of the appropriate type for the table schema.

- **attributes_to_get** (list) – A list of attribute names. If supplied, only the specified attribute names will be returned. Otherwise, all attributes will be returned.

submit()

RDS

boto.rds

class boto.rds.**RDSConnection**(*aws_access_key_id=None*, *aws_secret_access_key=None*, *is_secure=True*, *port=None*, *proxy=None*, *proxy_port=None*, *proxy_user=None*, *proxy_pass=None*, *debug=0*, *https_connection_factory=None*, *region=None*, *path='/'*)

APIVersion = '2011-04-01'

DefaultRegionEndpoint = 'rds.us-east-1.amazonaws.com'

DefaultRegionName = 'us-east-1'

authorize_dbsecurity_group(*group_name*, *cidr_ip=None*, *ec2_security_group_name=None*, *ec2_security_group_owner_id=None*)

Add a new rule to an existing security group. You need to pass in either src_security_group_name and src_security_group_owner_id OR a CIDR block but not both.

Parameters

- **group_name** (*string*) – The name of the security group you are adding the rule to.
- **ec2_security_group_name** (*string*) – The name of the EC2 security group you are granting access to.
- **ec2_security_group_owner_id** (*string*) – The ID of the owner of the EC2 security group you are granting access to.
- **cidr_ip** (*string*) – The CIDR block you are providing access to. See http://en.wikipedia.org/wiki/Classless_Inter-Domain_Routing

Return type bool

Returns True if successful.

create_dbinstance(*id*, *allocated_storage*, *instance_class*, *master_username*, *master_password*, *port=3306*, *engine='MySQL5.1'*, *db_name=None*, *param_group=None*, *security_groups=None*, *availability_zone=None*, *preferred_maintenance_window=None*, *backup_retention_period=None*, *preferred_backup_window=None*, *multi_az=False*, *engine_version=None*, *auto_minor_version_upgrade=True*)

Create a new DBInstance.

Parameters

- **id** (*str*) – Unique identifier for the new instance. Must contain 1-63 alphanumeric characters. First character must be a letter. May not end with a hyphen or contain two consecutive hyphens
- **allocated_storage** (*int*) – Initially allocated storage size, in GBs. Valid values are [5-1024]

- **instance_class** (*str*) – The compute and memory capacity of the DBInstance. Valid values are:
  - db.m1.small
  - db.m1.large
  - db.m1.xlarge
  - db.m2.xlarge
  - db.m2.2xlarge
  - db.m2.4xlarge
- **engine** (*str*) – Name of database engine. Must be MySQL5.1 for now.
- **master_username** (*str*) – Name of master user for the DBInstance. Must be 1-15 alphanumeric characters, first must be a letter.
- **master_password** (*str*) – Password of master user for the DBInstance. Must be 4-16 alphanumeric characters.
- **port** (*int*) – Port number on which database accepts connections. Valid values [1115-65535]. Defaults to 3306.
- **db_name** (*str*) – Name of a database to create when the DBInstance is created. Default is to create no databases.
- **param_group** (*str*) – Name of DBParameterGroup to associate with this DBInstance. If no groups are specified no parameter groups will be used.
- **security_groups** (*list of str or list of DBSecurityGroup objects*) – List of names of DBSecurityGroup to authorize on this DBInstance.
- **availability_zone** (*str*) – Name of the availability zone to place DBInstance into.
- **preferred_maintenance_window** (*str*) – The weekly time range (in UTC) during which maintenance can occur. Default is Sun:05:00-Sun:09:00
- **backup_retention_period** (*int*) – The number of days for which automated backups are retained. Setting this to zero disables automated backups.
- **preferred_backup_window** (*str*) – The daily time range during which automated backups are created (if enabled). Must be in h24:mi-hh24:mi format (UTC).
- **multi_az** (*bool*) – If True, specifies the DB Instance will be deployed in multiple availability zones.
- **engine_version** (*str*) – Version number of the database engine to use.
- **auto_minor_version_upgrade** (*bool*) – Indicates that minor engine upgrades will be applied automatically to the Read Replica during the maintenance window. Default is True.

Return type *boto.rds.dbinstance.DBInstance*

Returns The new db instance.

create_dbinstance_read_replica (*id*, *source_id*, *instance_class=None*, *port=3306*, *availability_zone=None*, *auto_minor_version_upgrade=None*)

Create a new DBInstance Read Replica.

Parameters

- **id** (*str*) – Unique identifier for the new instance. Must contain 1-63 alphanumeric characters. First character must be a letter. May not end with a hyphen or contain two consecutive hyphens
- **source_id** (*str*) – Unique identifier for the DB Instance for which this DB Instance will act as a Read Replica.
- **instance_class** (*str*) – The compute and memory capacity of the DBInstance. Default is to inherit from the source DB Instance.

  Valid values are:

  – db.m1.small

  – db.m1.large

  – db.m1.xlarge

  – db.m2.xlarge

  – db.m2.2xlarge

  – db.m2.4xlarge
- **port** (*int*) – Port number on which database accepts connections. Default is to inherit from source DB Instance. Valid values [1115-65535]. Defaults to 3306.
- **availability_zone** (*str*) – Name of the availability zone to place DBInstance into.
- **auto_minor_version_upgrade** (*bool*) – Indicates that minor engine upgrades will be applied automatically to the Read Replica during the maintenance window. Default is to inherit this value from the source DB Instance.

Return type *boto.rds.dbinstance.DBInstance*

Returns The new db instance.

create_dbsecurity_group (*name*, *description=None*)

Create a new security group for your account. This will create the security group within the region you are currently connected to.

Parameters

- **name** (*string*) – The name of the new security group
- **description** (*string*) – The description of the new security group

Return type *boto.rds.dbsecuritygroup.DBSecurityGroup*

Returns The newly created DBSecurityGroup

create_dbsnapshot (*snapshot_id*, *dbinstance_id*)

Create a new DB snapshot.

Parameters

- **snapshot_id** (*string*) – The identifier for the DBSnapshot
- **dbinstance_id** (*string*) – The source identifier for the RDS instance from which the snapshot is created.

Return type *boto.rds.dbsnapshot.DBSnapshot*

Returns The newly created DBSnapshot

create_parameter_group (*name*, *engine='MySQL5.1'*, *description=''*)

Create a new dbparameter group for your account.

Parameters

- **name** (*string*) – The name of the new dbparameter group
- **engine** (*str*) – Name of database engine.
- **description** (*string*) – The description of the new security group

Return type *boto.rds.dbsecuritygroup.DBSecurityGroup*

Returns The newly created DBSecurityGroup

delete_dbinstance(*id*, *skip_final_snapshot=False*, *final_snapshot_id=''*)
Delete an existing DBInstance.

Parameters

- **id** (*str*) – Unique identifier for the new instance.
- **skip_final_snapshot** (*bool*) – This parameter determines whether a final db snapshot is created before the instance is deleted. If True, no snapshot is created. If False, a snapshot is created before deleting the instance.
- **final_snapshot_id** (*str*) – If a final snapshot is requested, this is the identifier used for that snapshot.

Return type *boto.rds.dbinstance.DBInstance*

Returns The deleted db instance.

delete_dbsecurity_group(*name*)
Delete a DBSecurityGroup from your account.

Parameters **key_name** (*string*) – The name of the DBSecurityGroup to delete

delete_dbsnapshot(*identifier*)
Delete a DBSnapshot

Parameters **identifier** (*string*) – The identifier of the DBSnapshot to delete

delete_parameter_group(*name*)
Delete a DBSecurityGroup from your account.

Parameters **key_name** (*string*) – The name of the DBSecurityGroup to delete

get_all_dbinstances(*instance_id=None*, *max_records=None*, *marker=None*)
Retrieve all the DBInstances in your account.

Parameters

- **instance_id** (*str*) – DB Instance identifier. If supplied, only information this instance will be returned. Otherwise, info about all DB Instances will be returned.
- **max_records** (*int*) – The maximum number of records to be returned. If more results are available, a MoreToken will be returned in the response that can be used to retrieve additional records. Default is 100.
- **marker** (*str*) – The marker provided by a previous request.

Return type *list*

Returns A list of *boto.rds.dbinstance.DBInstance*

get_all_dbparameter_groups(*groupname=None*, *max_records=None*, *marker=None*)
Get all parameter groups associated with your account in a region.

Parameters

- **groupname** (*str*) – The name of the DBParameter group to retrieve. If not provided, all DBParameter groups will be returned.
- **max_records** (*int*) – The maximum number of records to be returned. If more results are available, a MoreToken will be returned in the response that can be used to retrieve additional records. Default is 100.
- **marker** (*str*) – The marker provided by a previous request.

Return type *list*

Returns A list of `boto.ec2.parametergroup.ParameterGroup`

get_all_dbparameters(*groupname*, *source=None*, *max_records=None*, *marker=None*)

Get all parameters associated with a ParameterGroup

Parameters

- **groupname** (*str*) – The name of the DBParameter group to retrieve.
- **source** (*str*) – Specifies which parameters to return. If not specified, all parameters will be returned. Valid values are: user|system|engine-default
- **max_records** (*int*) – The maximum number of records to be returned. If more results are available, a MoreToken will be returned in the response that can be used to retrieve additional records. Default is 100.
- **marker** (*str*) – The marker provided by a previous request.

Return type `boto.ec2.parametergroup.ParameterGroup`

Returns The ParameterGroup

get_all_dbsecurity_groups(*groupname=None*, *max_records=None*, *marker=None*)

Get all security groups associated with your account in a region.

Parameters

- **groupnames** (*list*) – A list of the names of security groups to retrieve. If not provided, all security groups will be returned.
- **max_records** (*int*) – The maximum number of records to be returned. If more results are available, a MoreToken will be returned in the response that can be used to retrieve additional records. Default is 100.
- **marker** (*str*) – The marker provided by a previous request.

Return type *list*

Returns A list of *boto.rds.dbsecuritygroup.DBSecurityGroup*

get_all_dbsnapshots(*snapshot_id=None*, *instance_id=None*, *max_records=None*, *marker=None*)

Get information about DB Snapshots.

Parameters

- **snapshot_id** (*str*) – The unique identifier of an RDS snapshot. If not provided, all RDS snapshots will be returned.
- **instance_id** (*str*) – The identifier of a DBInstance. If provided, only the DBSnapshots related to that instance will be returned. If not provided, all RDS snapshots will be returned.

- **max_records** (*int*) – The maximum number of records to be returned. If more results are available, a MoreToken will be returned in the response that can be used to retrieve additional records. Default is 100.
- **marker** (*str*) – The marker provided by a previous request.

Return type *list*

Returns A list of *boto.rds.dbsnapshot.DBSnapshot*

get_all_events(*source_identifier=None*, *source_type=None*, *start_time=None*, *end_time=None*, *max_records=None*, *marker=None*)
Get information about events related to your DBInstances, DBSecurityGroups and DBParameterGroups.

Parameters

- **source_identifier** (*str*) – If supplied, the events returned will be limited to those that apply to the identified source. The value of this parameter depends on the value of source_type. If neither parameter is specified, all events in the time span will be returned.
- **source_type** (*str*) – Specifies how the source_identifier should be interpreted. Valid values are: b-instance | db-security-group | db-parameter-group | db-snapshot
- **start_time** (*datetime*) – The beginning of the time interval for events. If not supplied, all available events will be returned.
- **end_time** (*datetime*) – The ending of the time interval for events. If not supplied, all available events will be returned.
- **max_records** (*int*) – The maximum number of records to be returned. If more results are available, a MoreToken will be returned in the response that can be used to retrieve additional records. Default is 100.
- **marker** (*str*) – The marker provided by a previous request.

Return type *list*

Returns A list of class:*boto.rds.event.Event*

modify_dbinstance(*id*, *param_group=None*, *security_groups=None*, *preferred_maintenance_window=None*, *master_password=None*, *allocated_storage=None*, *instance_class=None*, *backup_retention_period=None*, *preferred_backup_window=None*, *multi_az=False*, *apply_immediately=False*)
Modify an existing DBInstance.

Parameters

- **id** (*str*) – Unique identifier for the new instance.
- **security_groups** (*list of str or list of DBSecurityGroup objects*) – List of names of DBSecurityGroup to authorize on this DBInstance.
- **preferred_maintenance_window** (*str*) – The weekly time range (in UTC) during which maintenance can occur. Default is Sun:05:00-Sun:09:00
- **master_password** (*str*) – Password of master user for the DBInstance. Must be 4-15 alphanumeric characters.
- **allocated_storage** (*int*) – The new allocated storage size, in GBs. Valid values are [5-1024]
- **instance_class** (*str*) – The compute and memory capacity of the DBInstance. Changes will be applied at next maintenance window unless apply_immediately is True.

Valid values are:

– db.m1.small

– db.m1.large

– db.m1.xlarge

– db.m2.xlarge

– db.m2.2xlarge

– db.m2.4xlarge

- **apply_immediately** (*bool*) – If true, the modifications will be applied as soon as possible rather than waiting for the next preferred maintenance window.
- **backup_retention_period** (*int*) – The number of days for which automated backups are retained. Setting this to zero disables automated backups.
- **preferred_backup_window** (*str*) – The daily time range during which automated backups are created (if enabled). Must be in h24:mi-hh24:mi format (UTC).
- **multi_az** (*bool*) – If True, specifies the DB Instance will be deployed in multiple availability zones.

Return type *boto.rds.dbinstance.DBInstance*

Returns The modified db instance.

modify_parameter_group(*name*, *parameters=None*)

Modify a parameter group for your account.

Parameters

- **name** (*string*) – The name of the new parameter group
- **parameters** (list of *boto.rds.parametergroup.Parameter*) – The new parameters

Return type *boto.rds.parametergroup.ParameterGroup*

Returns The newly created ParameterGroup

reboot_dbinstance(*id*)

Reboot DBInstance.

Parameters **id** (*str*) – Unique identifier of the instance.

Return type *boto.rds.dbinstance.DBInstance*

Returns The rebooting db instance.

reset_parameter_group(*name*, *reset_all_params=False*, *parameters=None*)

Resets some or all of the parameters of a ParameterGroup to the default value

Parameters

- **key_name** (*string*) – The name of the ParameterGroup to reset
- **parameters** (list of *boto.rds.parametergroup.Parameter*) – The parameters to reset. If not supplied, all parameters will be reset.

restore_dbinstance_from_dbsnapshot(*identifier*, *instance_id*, *instance_class*, *port=None*, *availability_zone=None*)

Create a new DBInstance from a DB snapshot.

Parameters

- **identifier** (*string*) – The identifier for the DBSnapshot
- **instance_id** (*string*) – The source identifier for the RDS instance from which the snapshot is created.
- **instance_class** (*str*) – The compute and memory capacity of the DBInstance. Valid values are: db.m1.small | db.m1.large | db.m1.xlarge | db.m2.2xlarge | db.m2.4xlarge
- **port** (*int*) – Port number on which database accepts connections. Valid values [1115-65535]. Defaults to 3306.
- **availability_zone** (*str*) – Name of the availability zone to place DBInstance into.

Return type *boto.rds.dbinstance.DBInstance*

Returns The newly created DBInstance

restore_dbinstance_from_point_in_time(*source_instance_id*, *target_instance_id*, *use_latest=False*, *restore_time=None*, *dbinstance_class=None*, *port=None*, *availability_zone=None*)

Create a new DBInstance from a point in time.

Parameters

- **source_instance_id** (*string*) – The identifier for the source DBInstance.
- **target_instance_id** (*string*) – The identifier of the new DBInstance.
- **use_latest** (*bool*) – If True, the latest snapshot availabile will be used.
- **restore_time** (*datetime*) – The date and time to restore from. Only used if use_latest is False.
- **instance_class** (*str*) – The compute and memory capacity of the DBInstance. Valid values are: db.m1.small | db.m1.large | db.m1.xlarge | db.m2.2xlarge | db.m2.4xlarge
- **port** (*int*) – Port number on which database accepts connections. Valid values [1115-65535]. Defaults to 3306.
- **availability_zone** (*str*) – Name of the availability zone to place DBInstance into.

Return type *boto.rds.dbinstance.DBInstance*

Returns The newly created DBInstance

revoke_dbsecurity_group(*group_name*, *ec2_security_group_name=None*, *ec2_security_group_owner_id=None*, *cidr_ip=None*)

Remove an existing rule from an existing security group. You need to pass in either ec2_security_group_name and ec2_security_group_owner_id OR a CIDR block.

Parameters

- **group_name** (*string*) – The name of the security group you are removing the rule from.
- **ec2_security_group_name** (*string*) – The name of the EC2 security group from which you are removing access.
- **ec2_security_group_owner_id** (*string*) – The ID of the owner of the EC2 security from which you are removing access.
- **cidr_ip** (*string*) – The CIDR block from which you are removing access. See http://en.wikipedia.org/wiki/Classless_Inter-Domain_Routing

Return type bool

Returns True if successful.

revoke_security_group (*group_name*, *ec2_security_group_name=None*, *ec2_security_group_owner_id=None*, *cidr_ip=None*)

Remove an existing rule from an existing security group. You need to pass in either ec2_security_group_name and ec2_security_group_owner_id OR a CIDR block.

Parameters

- **group_name** (*string*) – The name of the security group you are removing the rule from.
- **ec2_security_group_name** (*string*) – The name of the EC2 security group from which you are removing access.
- **ec2_security_group_owner_id** (*string*) – The ID of the owner of the EC2 security from which you are removing access.
- **cidr_ip** (*string*) – The CIDR block from which you are removing access. See http://en.wikipedia.org/wiki/Classless_Inter-Domain_Routing

Return type bool

Returns True if successful.

boto.rds.**connect_to_region** (*region_name*, ***kw_params*)

Given a valid region name, return a `boto.ec2.connection.EC2Connection`. Any additional parameters after the region_name are passed on to the connect method of the region object.

Type str

Parameters **region_name** – The name of the region to connect to.

Return type `boto.ec2.connection.EC2Connection` or `None`

Returns A connection to the given region, or None if an invalid region name is given

boto.rds.**regions** ()

Get all available regions for the RDS service.

Return type *list*

Returns A list of `boto.rds.regioninfo.RDSRegionInfo`

boto.rds.dbinstance

class boto.rds.dbinstance.**DBInstance** (*connection=None*, *id=None*)

Represents a RDS DBInstance

endElement (*name*, *value*, *connection*)

modify (*param_group=None*, *security_groups=None*, *preferred_maintenance_window=None*, *master_password=None*, *allocated_storage=None*, *instance_class=None*, *backup_retention_period=None*, *preferred_backup_window=None*, *multi_az=False*, *apply_immediately=False*)

Modify this DBInstance.

Parameters

- **security_groups** (*list of str or list of DBSecurityGroup objects*) – List of names of DBSecurityGroup to authorize on this DBInstance.
- **preferred_maintenance_window** (*str*) – The weekly time range (in UTC) during which maintenance can occur. Default is Sun:05:00-Sun:09:00

- **master_password** (*str*) – Password of master user for the DBInstance. Must be 4-15 alphanumeric characters.
- **allocated_storage** (*int*) – The new allocated storage size, in GBs. Valid values are [5-1024]
- **instance_class** (*str*) – The compute and memory capacity of the DBInstance. Changes will be applied at next maintenance window unless apply_immediately is True.

  Valid values are:

  – db.m1.small

  – db.m1.large

  – db.m1.xlarge

  – db.m2.xlarge

  – db.m2.2xlarge

  – db.m2.4xlarge
- **apply_immediately** (*bool*) – If true, the modifications will be applied as soon as possible rather than waiting for the next preferred maintenance window.
- **backup_retention_period** (*int*) – The number of days for which automated backups are retained. Setting this to zero disables automated backups.
- **preferred_backup_window** (*str*) – The daily time range during which automated backups are created (if enabled). Must be in h24:mi-hh24:mi format (UTC).
- **multi_az** (*bool*) – If True, specifies the DB Instance will be deployed in multiple availability zones.

Return type *boto.rds.dbinstance.DBInstance*

Returns The modified db instance.

reboot()
Reboot this DBInstance

Return type *boto.rds.dbsnapshot.DBSnapshot*

Returns The newly created DBSnapshot

snapshot(*snapshot_id*)
Create a new DB snapshot of this DBInstance.

Parameters **identifier** (*string*) – The identifier for the DBSnapshot

Return type *boto.rds.dbsnapshot.DBSnapshot*

Returns The newly created DBSnapshot

startElement(*name*, *attrs*, *connection*)

stop(*skip_final_snapshot=False*, *final_snapshot_id=''*)
Delete this DBInstance.

Parameters

- **skip_final_snapshot** (*bool*) – This parameter determines whether a final db snapshot is created before the instance is deleted. If True, no snapshot is created. If False, a snapshot is created before deleting the instance.

- **final_snapshot_id** (*str*) – If a final snapshot is requested, this is the identifier used for that snapshot.

Return type *boto.rds.dbinstance.DBInstance*

Returns The deleted db instance.

update (*validate=False*)
Update the DB instance's status information by making a call to fetch the current instance attributes from the service.

Parameters validate (*bool*) – By default, if EC2 returns no data about the instance the update method returns quietly. If the validate param is True, however, it will raise a ValueError exception if no data is returned from EC2.

class boto.rds.dbinstance.**PendingModifiedValues**

endElement (*name*, *value*, *connection*)

startElement (*name*, *attrs*, *connection*)

boto.rds.dbsecuritygroup

Represents an DBSecurityGroup

class boto.rds.dbsecuritygroup.**DBSecurityGroup** (*connection=None*, *owner_id=None*, *name=None*, *description=None*)

authorize (*cidr_ip=None*, *ec2_group=None*)
Add a new rule to this DBSecurity group. You need to pass in either a CIDR block to authorize or and EC2 SecurityGroup.

@type cidr_ip: string @param cidr_ip: A valid CIDR IP range to authorize

@type ec2_group: `boto.ec2.securitygroup.SecurityGroup>`

@rtype: bool @return: True if successful.

delete ()

endElement (*name*, *value*, *connection*)

revoke (*cidr_ip=None*, *ec2_group=None*)
Revoke access to a CIDR range or EC2 SecurityGroup. You need to pass in either a CIDR block or an EC2 SecurityGroup from which to revoke access.

@type cidr_ip: string @param cidr_ip: A valid CIDR IP range to revoke

@type ec2_group: `boto.ec2.securitygroup.SecurityGroup>`

@rtype: bool @return: True if successful.

startElement (*name*, *attrs*, *connection*)

class boto.rds.dbsecuritygroup.**EC2SecurityGroup** (*parent=None*)

endElement (*name*, *value*, *connection*)

startElement (*name*, *attrs*, *connection*)

class boto.rds.dbsecuritygroup.**IPRange** (*parent=None*)

endElement (*name*, *value*, *connection*)

startElement (*name*, *attrs*, *connection*)

boto.rds.dbsnapshot

class boto.rds.dbsnapshot.**DBSnapshot** (*connection=None*, *id=None*)
Represents a RDS DB Snapshot

endElement (*name*, *value*, *connection*)

startElement (*name*, *attrs*, *connection*)

boto.rds.event

class boto.rds.event.**Event** (*connection=None*)

endElement (*name*, *value*, *connection*)

startElement (*name*, *attrs*, *connection*)

boto.rds.parametergroup

class boto.rds.parametergroup.**Parameter** (*group=None*, *name=None*)
Represents a RDS Parameter

ValidApplyMethods = ['immediate', 'pending-reboot']

ValidApplyTypes = ['static', 'dynamic']

ValidSources = ['user', 'system', 'engine-default']

ValidTypes = {'integer': <type 'int'>, 'boolean': <type 'bool'>, 'string': <type 'str'>}

apply (*immediate=False*)

endElement (*name*, *value*, *connection*)

get_value ()

merge (*d*, *i*)

set_value (*value*)

startElement (*name*, *attrs*, *connection*)

value

class boto.rds.parametergroup.**ParameterGroup** (*connection=None*)

add_param (*name*, *value*, *apply_method*)

endElement (*name*, *value*, *connection*)

get_params ()

modifiable ()

startElement (*name*, *attrs*, *connection*)

cloudformation

boto.cloudformation

boto.cloudformation.stack

class `boto.cloudformation.stack.`**`Output`**(*connection=None*)

> **`endElement`**(*name, value, connection*)
>
> **`startElement`**(*name, attrs, connection*)

class `boto.cloudformation.stack.`**`Parameter`**(*connection=None*)

> **`endElement`**(*name, value, connection*)
>
> **`startElement`**(*name, attrs, connection*)

class `boto.cloudformation.stack.`**`Stack`**(*connection=None*)

> **`delete`**()
>
> **`describe_events`**(*next_token=None*)
>
> **`describe_resource`**(*logical_resource_id*)
>
> **`describe_resources`**(*logical_resource_id=None, physical_resource_id=None*)
>
> **`endElement`**(*name, value, connection*)
>
> **`get_template`**()
>
> **`list_resources`**(*next_token=None*)
>
> **`startElement`**(*name, attrs, connection*)
>
> **`update`**()

class `boto.cloudformation.stack.`**`StackEvent`**(*connection=None*)

> **`endElement`**(*name, value, connection*)
>
> **`startElement`**(*name, attrs, connection*)
>
> **`valid_states`** **= ('CREATE_IN_PROGRESS', 'CREATE_FAILED', 'CREATE_COMPLETE', 'DELETE_IN_PROG**

class `boto.cloudformation.stack.`**`StackResource`**(*connection=None*)

> **`endElement`**(*name, value, connection*)
>
> **`startElement`**(*name, attrs, connection*)

class `boto.cloudformation.stack.`**`StackResourceSummary`**(*connection=None*)

> **`endElement`**(*name, value, connection*)
>
> **`startElement`**(*name, attrs, connection*)

class `boto.cloudformation.stack.`**`StackSummary`**(*connection=None*)

endElement (*name*, *value*, *connection*)

startElement (*name*, *attrs*, *connection*)

boto.cloudformation.template

class boto.cloudformation.template.**Template** (*connection=None*)

endElement (*name*, *value*, *connection*)

startElement (*name*, *attrs*, *connection*)

class boto.cloudformation.template.**TemplateParameter** (*parent*)

endElement (*name*, *value*, *connection*)

startElement (*name*, *attrs*, *connection*)

IAM

boto.iam

boto.iam.connection

class boto.iam.connection.**IAMConnection** (*aws_access_key_id=None*, *aws_secret_access_key=None*, *is_secure=True*, *port=None*, *proxy=None*, *proxy_port=None*, *proxy_user=None*, *proxy_pass=None*, *host='iam.amazonaws.com'*, *debug=0*, *https_connection_factory=None*, *path='/'*)

APIVersion = '2010-05-08'

add_user_to_group (*group_name*, *user_name*)

Add a user to a group

Parameters

- **group_name** (*string*) – The name of the group
- **user_name** (*string*) – The to be added to the group.

create_access_key (*user_name=None*)

Create a new AWS Secret Access Key and corresponding AWS Access Key ID for the specified user. The default status for new keys is Active

If the user_name is not specified, the user_name is determined implicitly based on the AWS Access Key ID used to sign the request.

Parameters user_name (*string*) – The username of the user

create_account_alias (*alias*)

Creates a new alias for the AWS account.

For more information on account id aliases, please see http://goo.gl/ToB7G

Parameters alias (*string*) – The alias to attach to the account.

create_group (*group_name*, *path='/'*)
Create a group.

Parameters

- **group_name** (*string*) – The name of the new group
- **path** (*string*) – The path to the group (Optional). Defaults to /.

create_login_profile (*user_name*, *password*)
Creates a login profile for the specified user, give the user the ability to access AWS services and the AWS Management Console.

Parameters

- **user_name** (*string*) – The name of the user
- **password** (*string*) – The new password for the user

create_user (*user_name*, *path='/'*)
Create a user.

Parameters

- **user_name** (*string*) – The name of the new user
- **path** (*string*) – The path in which the user will be created. Defaults to /.

deactivate_mfa_device (*user_name*, *serial_number*)
Deactivates the specified MFA device and removes it from association with the user.

Parameters

- **user_name** (*string*) – The username of the user
- **seriasl_number** – The serial number which uniquely identifies the MFA device.

delete_access_key (*access_key_id*, *user_name=None*)
Delete an access key associated with a user.

If the user_name is not specified, it is determined implicitly based on the AWS Access Key ID used to sign the request.

Parameters

- **access_key_id** (*string*) – The ID of the access key to be deleted.
- **user_name** (*string*) – The username of the user

delete_account_alias (*alias*)
Deletes an alias for the AWS account.

For more information on account id aliases, please see http://goo.gl/ToB7G

Parameters **alias** (*string*) – The alias to remove from the account.

delete_group (*group_name*)
Delete a group. The group must not contain any Users or have any attached policies

Parameters **group_name** (*string*) – The name of the group to delete.

delete_group_policy (*group_name*, *policy_name*)
Deletes the specified policy document for the specified group.

Parameters

- **group_name** (*string*) – The name of the group the policy is associated with.

- **policy_name** (*string*) – The policy document to delete.

delete_login_profile(*user_name*)

Deletes the login profile associated with the specified user.

Parameters **user_name** (*string*) – The name of the user to delete.

delete_server_cert(*cert_name*)

Delete the specified server certificate.

Parameters **cert_name** (*string*) – The name of the server certificate you want to delete.

delete_signing_cert(*cert_id*, *user_name=None*)

Delete a signing certificate associated with a user.

If the user_name is not specified, it is determined implicitly based on the AWS Access Key ID used to sign the request.

Parameters

- **user_name** (*string*) – The username of the user
- **cert_id** (*string*) – The ID of the certificate.

delete_user(*user_name*)

Delete a user including the user's path, GUID and ARN.

If the user_name is not specified, the user_name is determined implicitly based on the AWS Access Key ID used to sign the request.

Parameters **user_name** (*string*) – The name of the user to delete.

delete_user_policy(*user_name*, *policy_name*)

Deletes the specified policy document for the specified user.

Parameters

- **user_name** (*string*) – The name of the user the policy is associated with.
- **policy_name** (*string*) – The policy document to delete.

enable_mfa_device(*user_name*, *serial_number*, *auth_code_1*, *auth_code_2*)

Enables the specified MFA device and associates it with the specified user.

Parameters

- **user_name** (*string*) – The username of the user
- **seriasl_number** – The serial number which uniquely identifies the MFA device.
- **auth_code_1** (*string*) – An authentication code emitted by the device.
- **auth_code_2** (*string*) – A subsequent authentication code emitted by the device.

get_account_alias()

Get the alias for the current account.

This is referred to in the docs as list_account_aliases, but it seems you can only have one account alias currently.

For more information on account id aliases, please see http://goo.gl/ToB7G

get_account_summary()

Get the alias for the current account.

This is referred to in the docs as list_account_aliases, but it seems you can only have one account alias currently.

For more information on account id aliases, please see http://goo.gl/ToB7G

get_all_access_keys(*user_name*, *marker=None*, *max_items=None*)
Get all access keys associated with an account.

Parameters

- **user_name** (*string*) – The username of the user
- **marker** (*string*) – Use this only when paginating results and only in follow-up request after you've received a response where the results are truncated. Set this to the value of the Marker element in the response you just received.
- **max_items** (*int*) – Use this only when paginating results to indicate the maximum number of groups you want in the response.

get_all_group_policies(*group_name*, *marker=None*, *max_items=None*)
List the names of the policies associated with the specified group.

Parameters

- **group_name** (*string*) – The name of the group the policy is associated with.
- **marker** (*string*) – Use this only when paginating results and only in follow-up request after you've received a response where the results are truncated. Set this to the value of the Marker element in the response you just received.
- **max_items** (*int*) – Use this only when paginating results to indicate the maximum number of groups you want in the response.

get_all_groups(*path_prefix='/'*, *marker=None*, *max_items=None*)
List the groups that have the specified path prefix.

Parameters

- **path_prefix** (*string*) – If provided, only groups whose paths match the provided prefix will be returned.
- **marker** (*string*) – Use this only when paginating results and only in follow-up request after you've received a response where the results are truncated. Set this to the value of the Marker element in the response you just received.
- **max_items** (*int*) – Use this only when paginating results to indicate the maximum number of groups you want in the response.

get_all_mfa_devices(*user_name*, *marker=None*, *max_items=None*)
Get all MFA devices associated with an account.

Parameters

- **user_name** (*string*) – The username of the user
- **marker** (*string*) – Use this only when paginating results and only in follow-up request after you've received a response where the results are truncated. Set this to the value of the Marker element in the response you just received.
- **max_items** (*int*) – Use this only when paginating results to indicate the maximum number of groups you want in the response.

get_all_server_certs(*path_prefix='/'*, *marker=None*, *max_items=None*)
Lists the server certificates that have the specified path prefix. If none exist, the action returns an empty list.

Parameters

- **path_prefix** (*string*) – If provided, only certificates whose paths match the provided prefix will be returned.
- **marker** (*string*) – Use this only when paginating results and only in follow-up request after you've received a response where the results are truncated. Set this to the value of the Marker element in the response you just received.
- **max_items** (*int*) – Use this only when paginating results to indicate the maximum number of groups you want in the response.

get_all_signing_certs (*marker=None*, *max_items=None*, *user_name=None*)
Get all signing certificates associated with an account.

If the user_name is not specified, it is determined implicitly based on the AWS Access Key ID used to sign the request.

Parameters

- **marker** (*string*) – Use this only when paginating results and only in follow-up request after you've received a response where the results are truncated. Set this to the value of the Marker element in the response you just received.
- **max_items** (*int*) – Use this only when paginating results to indicate the maximum number of groups you want in the response.
- **user_name** (*string*) – The username of the user

get_all_user_policies (*user_name*, *marker=None*, *max_items=None*)
List the names of the policies associated with the specified user.

Parameters

- **user_name** (*string*) – The name of the user the policy is associated with.
- **marker** (*string*) – Use this only when paginating results and only in follow-up request after you've received a response where the results are truncated. Set this to the value of the Marker element in the response you just received.
- **max_items** (*int*) – Use this only when paginating results to indicate the maximum number of groups you want in the response.

get_all_users (*path_prefix='/'*, *marker=None*, *max_items=None*)
List the users that have the specified path prefix.

Parameters

- **path_prefix** (*string*) – If provided, only users whose paths match the provided prefix will be returned.
- **marker** (*string*) – Use this only when paginating results and only in follow-up request after you've received a response where the results are truncated. Set this to the value of the Marker element in the response you just received.
- **max_items** (*int*) – Use this only when paginating results to indicate the maximum number of groups you want in the response.

get_group (*group_name*, *marker=None*, *max_items=None*)
Return a list of users that are in the specified group.

Parameters

- **group_name** (*string*) – The name of the group whose information should be returned.

- **marker** (*string*) – Use this only when paginating results and only in follow-up request after you've received a response where the results are truncated. Set this to the value of the Marker element in the response you just received.
- **max_items** (*int*) – Use this only when paginating results to indicate the maximum number of groups you want in the response.

get_group_policy(*group_name*, *policy_name*)
Retrieves the specified policy document for the specified group.

Parameters

- **group_name** (*string*) – The name of the group the policy is associated with.
- **policy_name** (*string*) – The policy document to get.

get_groups_for_user(*user_name*, *marker=None*, *max_items=None*)
List the groups that a specified user belongs to.

Parameters

- **user_name** (*string*) – The name of the user to list groups for.
- **marker** (*string*) – Use this only when paginating results and only in follow-up request after you've received a response where the results are truncated. Set this to the value of the Marker element in the response you just received.
- **max_items** (*int*) – Use this only when paginating results to indicate the maximum number of groups you want in the response.

get_login_profiles(*user_name*)
Retrieves the login profile for the specified user.

Parameters **user_name** (*string*) – The username of the user

get_response(*action*, *params*, *path='/'*, *parent=None*, *verb='GET'*, *list_marker='Set'*)
Utility method to handle calls to IAM and parsing of responses.

get_server_certificate(*cert_name*)
Retrieves information about the specified server certificate.

Parameters **cert_name** (*string*) – The name of the server certificate you want to retrieve information about.

get_signin_url(*service='ec2'*)
Get the URL where IAM users can use their login profile to sign in to this account's console.

Parameters **service** (*string*) – Default service to go to in the console.

get_user(*user_name=None*)
Retrieve information about the specified user.

If the user_name is not specified, the user_name is determined implicitly based on the AWS Access Key ID used to sign the request.

Parameters **user_name** (*string*) – The name of the user to delete. If not specified, defaults to user making request.

get_user_policy(*user_name*, *policy_name*)
Retrieves the specified policy document for the specified user.

Parameters

- **user_name** (*string*) – The name of the user the policy is associated with.
- **policy_name** (*string*) – The policy document to get.

put_group_policy(*group_name*, *policy_name*, *policy_json*)
Adds or updates the specified policy document for the specified group.

Parameters

- **group_name** (*string*) – The name of the group the policy is associated with.
- **policy_name** (*string*) – The policy document to get.
- **policy_json** (*string*) – The policy document.

put_user_policy(*user_name*, *policy_name*, *policy_json*)
Adds or updates the specified policy document for the specified user.

Parameters

- **user_name** (*string*) – The name of the user the policy is associated with.
- **policy_name** (*string*) – The policy document to get.
- **policy_json** (*string*) – The policy document.

remove_user_from_group(*group_name*, *user_name*)
Remove a user from a group.

Parameters

- **group_name** (*string*) – The name of the group
- **user_name** (*string*) – The user to remove from the group.

resync_mfa_device(*user_name*, *serial_number*, *auth_code_1*, *auth_code_2*)
Syncronizes the specified MFA device with the AWS servers.

Parameters

- **user_name** (*string*) – The username of the user
- **seriasl_number** – The serial number which uniquely identifies the MFA device.
- **auth_code_1** (*string*) – An authentication code emitted by the device.
- **auth_code_2** (*string*) – A subsequent authentication code emitted by the device.

update_access_key(*access_key_id*, *status*, *user_name=None*)
Changes the status of the specified access key from Active to Inactive or vice versa. This action can be used to disable a user's key as part of a key rotation workflow.

If the user_name is not specified, the user_name is determined implicitly based on the AWS Access Key ID used to sign the request.

Parameters

- **access_key_id** (*string*) – The ID of the access key.
- **status** (*string*) – Either Active or Inactive.
- **user_name** (*string*) – The username of user (optional).

update_group(*group_name*, *new_group_name=None*, *new_path=None*)
Updates name and/or path of the specified group.

Parameters

- **group_name** (*string*) – The name of the new group
- **new_group_name** (*string*) – If provided, the name of the group will be changed to this name.

- **new_path** (*string*) – If provided, the path of the group will be changed to this path.

update_login_profile (*user_name*, *password*)
Resets the password associated with the user's login profile.

Parameters

- **user_name** (*string*) – The name of the user
- **password** (*string*) – The new password for the user

update_server_cert (*cert_name*, *new_cert_name=None*, *new_path=None*)
Updates the name and/or the path of the specified server certificate.

Parameters

- **cert_name** (*string*) – The name of the server certificate that you want to update.
- **new_cert_name** (*string*) – The new name for the server certificate. Include this only if you are updating the server certificate's name.
- **new_path** (*string*) – If provided, the path of the certificate will be changed to this path.

update_signing_cert (*cert_id*, *status*, *user_name=None*)
Change the status of the specified signing certificate from Active to Inactive or vice versa.

If the user_name is not specified, it is determined implicitly based on the AWS Access Key ID used to sign the request.

Parameters

- **cert_id** (*string*) – The ID of the signing certificate
- **status** (*string*) – Either Active or Inactive.
- **user_name** (*string*) – The username of the user

update_user (*user_name*, *new_user_name=None*, *new_path=None*)
Updates name and/or path of the specified user.

Parameters

- **user_name** (*string*) – The name of the user
- **new_user_name** (*string*) – If provided, the username of the user will be changed to this username.
- **new_path** (*string*) – If provided, the path of the user will be changed to this path.

upload_server_cert (*cert_name*, *cert_body*, *private_key*, *cert_chain=None*, *path=None*)
Uploads a server certificate entity for the AWS Account. The server certificate entity includes a public key certificate, a private key, and an optional certificate chain, which should all be PEM-encoded.

Parameters

- **cert_name** (*string*) – The name for the server certificate. Do not include the path in this value.
- **cert_body** (*string*) – The contents of the public key certificate in PEM-encoded format.
- **private_key** (*string*) – The contents of the private key in PEM-encoded format.
- **cert_chain** (*string*) – The contents of the certificate chain. This is typically a concatenation of the PEM-encoded public key certificates of the chain.

- **path** (*string*) – The path for the server certificate.

upload_signing_cert (*cert_body*, *user_name=None*)
Uploads an X.509 signing certificate and associates it with the specified user.

If the user_name is not specified, it is determined implicitly based on the AWS Access Key ID used to sign the request.

Parameters

- **cert_body** (*string*) – The body of the signing certificate.
- **user_name** (*string*) – The username of the user

boto.iam.summarymap

class boto.iam.summarymap.**SummaryMap** (*parent=None*)

endElement (*name*, *value*, *connection*)

startElement (*name*, *attrs*, *connection*)

An Introduction to boto's SQS interface

This tutorial focuses on the boto interface to the Simple Queue Service from Amazon Web Services. This tutorial assumes that you have boto already downloaded and installed.

Creating a Connection

The first step in accessing SQS is to create a connection to the service. There are two ways to do this in boto. The first is:

```
>>> from boto.sqs.connection import SQSConnection
>>> conn = SQSConnection('<aws access key>', '<aws secret key>')
```

At this point the variable conn will point to an SQSConnection object. Bear in mind that just as any other AWS service SQS is region-specfic. Also important to note is that by default, if no region is provided, it'll connect to the US-EAST-1 region. In this example, the AWS access key and AWS secret key are passed in to the method explicitely. Alternatively, you can set the environment variables:

AWS_ACCESS_KEY_ID - Your AWS Access Key ID AWS_SECRET_ACCESS_KEY - Your AWS Secret Access Key

and then call the constructor without any arguments, like this:

```
>>> conn = SQSConnection()
```

There is also a shortcut function in the boto package, called connect_sqs that may provide a slightly easier means of creating a connection:

```
>>> import boto
>>> conn = boto.connect_sqs()
```

In either case, conn will point to an SQSConnection object which we will use throughout the remainder of this tutorial.

Creating a Queue

Once you have a connection established with SQS, you will probably want to create a queue. In its simplest form, that can be accomplished as follows:

```
>>> q = conn.create_queue('myqueue')
```

The create_queue method will create (and return) the requested queue if it does not exist or will return the existing queue if it does. There is an optional parameter to create_queue called visibility_timeout. This basically controls how long a message will remain invisible to other queue readers once it has been read (see SQS documentation for more detailed explanation). If this is not explicitly specified the queue will be created with whatever default value SQS provides (currently 30 seconds). If you would like to specify another value, you could do so like this:

```
>>> q = conn.create_queue('myqueue', 120)
```

This would establish a default visibility timeout for this queue of 120 seconds. As you will see later on, this default value for the queue can also be overridden each time a message is read from the queue. If you want to check what the default visibility timeout is for a queue:

```
>>> q.get_timeout()
30
```

Listing all Queues

To retrieve a list of the queues for your account in the current region:

```
>>> conn.get_all_queues()
[
    Queue(https://queue.amazonaws.com/411358162645/myqueue),
    Queue(https://queue.amazonaws.com/411358162645/another_queue),
    Queue(https://queue.amazonaws.com/411358162645/another_queue2)
]
```

This will leave you with a list of all of your *`boto.sqs.queue.Queue`* instances. Alternatively, if you wanted to only list the queues that started with `'another'`:

```
>>> conn.get_all_queues(prefix='another')
[
    Queue(https://queue.amazonaws.com/411358162645/another_queue),
    Queue(https://queue.amazonaws.com/411358162645/another_queue2)
]
```

Getting a Queue (by name)

If you wish to explicitly retrieve an existing queue and the name of the queue is known, you can retrieve the queue as follows:

```
>>> my_queue = conn.get_queue('myqueue')
Queue(https://queue.amazonaws.com/411358162645/myqueue)
```

This leaves you with a single *`boto.sqs.queue.Queue`*, which abstracts the SQS Queue named 'myqueue'.

Writing Messages

Once you have a queue setup, presumably you will want to write some messages to it. SQS doesn't care what kind of information you store in your messages or what format you use to store it. As long as the amount of data per message is less than or equal to 256Kb, SQS won't complain.

So, first we need to create a Message object:

```
>>> from boto.sqs.message import Message
>>> m = Message()
>>> m.set_body('This is my first message.')
>>> status = q.write(m)
```

The write method returns a True if everything went well. If the write didn't succeed it will either return a False (meaning SQS simply chose not to write the message for some reason) or an exception if there was some sort of problem with the request.

Writing Messages (Custom Format)

The technique above will work only if you use boto's default Message payload format; however, you may have a lot of specific requirements around the format of the message data. For example, you may want to store one big string or you might want to store something that looks more like RFC822 messages or you might want to store a binary payload such as pickled Python objects.

The way boto deals with this issue is to define a simple Message object that treats the message data as one big string which you can set and get. If that Message object meets your needs, you're good to go. However, if you need to incorporate different behavior in your message or handle different types of data you can create your own Message class. You just need to register that class with the boto queue object so that it knows that, when you read a message from the queue, it should create one of your message objects rather than the default boto Message object. To register your message class, you would:

```
>>> import MyMessage
>>> q.set_message_class(MyMessage)
>>> m = MyMessage()
>>> m.set_body('This is my first message.')
>>> status = q.write(m)
```

where MyMessage is the class definition for your message class. Your message class should subclass the boto Message because there is a small bit of Python magic happening in the __setattr__ method of the boto Message class.

Reading Messages

So, now we have a message in our queue. How would we go about reading it? Here's one way:

```
>>> rs = q.get_messages()
>>> len(rs)
1
>>> m = rs[0]
>>> m.get_body()
u'This is my first message'
```

The get_messages method also returns a ResultSet object as described above. In addition to the special attributes that we already talked about the ResultSet object also contains any results returned by the request. To get at the results you can treat the ResultSet as a sequence object (e.g. a list). We can check the length (how many results) and access particular items within the list using the slice notation familiar to Python programmers.

At this point, we have read the message from the queue and SQS will make sure that this message remains invisible to other readers of the queue until the visibility timeout period for the queue expires. If you delete the message before the timeout period expires then no one else will ever see the message again. However, if you don't delete it (maybe because your reader crashed or failed in some way, for example) it will magically reappear in my queue for someone else to read. If you aren't happy with the default visibility timeout defined for the queue, you can override it when you read a message:

```
>>> q.get_messages(visibility_timeout=60)
```

This means that regardless of what the default visibility timeout is for the queue, this message will remain invisible to other readers for 60 seconds.

The get_messages method can also return more than a single message. By passing a num_messages parameter (defaults to 1) you can control the maximum number of messages that will be returned by the method. To show this feature off, first let's load up a few more messages.

```
>>> for i in range(1, 11):
...     m = Message()
...     m.set_body('This is message %d' % i)
...     q.write(m)
...
>>> rs = q.get_messages(10)
>>> len(rs)
10
```

Don't be alarmed if the length of the result set returned by the get_messages call is less than 10. Sometimes it takes some time for new messages to become visible in the queue. Give it a minute or two and they will all show up.

If you want a slightly simpler way to read messages from a queue, you can use the read method. It will either return the message read or it will return None if no messages were available. You can also pass a visibility_timeout parameter to read, if you desire:

```
>>> m = q.read(60)
>>> m.get_body()
u'This is my first message'
```

Deleting Messages and Queues

As stated above, messages are never deleted by the queue unless explicitly told to do so. To remove a message from a queue:

```
>>> q.delete_message(m)
[]
```

If I want to delete the entire queue, I would use:

```
>>> conn.delete_queue(q)
```

However, and this is a good safe guard, this won't succeed unless the queue is empty.

Additional Information

The above tutorial covers the basic operations of creating queues, writing messages, reading messages, deleting messages, and deleting queues. There are a few utility methods in boto that might be useful as well. For example, to count the number of messages in a queue:

```
>>> q.count()
10
```

This can be handy but is command as well as the other two utility methods I'll describe in a minute are inefficient and should be used with caution on queues with lots of messages (e.g. many hundreds or more). Similarly, you can clear (delete) all messages in a queue with:

```
>>> q.clear()
```

Be REAL careful with that one! Finally, if you want to dump all of the messages in a queue to a local file:

```
>>> q.dump('messages.txt', sep='\n------------------\n')
```

This will read all of the messages in the queue and write the bodies of each of the messages to the file messages.txt. The option sep argument is a separator that will be printed between each message body in the file.

SQS

boto.sqs

boto.sqs.**connect_to_region**(*region_name*, ***kw_params*)

boto.sqs.**regions**()
Get all available regions for the SQS service.

Return type *list*

Returns A list of `boto.ec2.regioninfo.RegionInfo`

boto.sqs.attributes

Represents an SQS Attribute Name/Value set

class boto.sqs.attributes.**Attributes**(*parent*)

endElement(*name*, *value*, *connection*)

startElement(*name*, *attrs*, *connection*)

boto.sqs.connection

class boto.sqs.connection.**SQSConnection**(*aws_access_key_id=None*, *aws_secret_access_key=None*, *is_secure=True*, *port=None*, *proxy=None*, *proxy_port=None*, *proxy_user=None*, *proxy_pass=None*, *debug=0*, *https_connection_factory=None*, *region=None*, *path='/'*, *security_token=None*)

A Connection to the SQS Service.

APIVersion = '2011-10-01'

DefaultContentType = 'text/plain'

DefaultRegionEndpoint = 'sqs.us-east-1.amazonaws.com'

`DefaultRegionName` = **'us-east-1'**

`ResponseError`
alias of `SQSError`

`add_permission`(*queue*, *label*, *aws_account_id*, *action_name*)
Add a permission to a queue.

Parameters

- **`queue`** (`boto.sqs.queue.Queue`) – The queue object
- **`label`** (*`str or unicode`*) – A unique identification of the permission you are setting. Maximum of 80 characters `[0-9a-zA-Z_-]` Example, AliceSendMessage
- **`principal_id`** – The AWS account number of the principal who will be given permission. The principal must have an AWS account, but does not need to be signed up for Amazon SQS. For information about locating the AWS account identification.
- **`action_name`** (*`str or unicode`*) – The action. Valid choices are: *|SendMessage|ReceiveMessage|DeleteMessage| ChangeMessageVisibility|GetQueueAttributes

Return type bool

Returns True if successful, False otherwise.

`change_message_visibility`(*queue*, *receipt_handle*, *visibility_timeout*)
Extends the read lock timeout for the specified message from the specified queue to the specified value.

Parameters

- **`queue`** (A `boto.sqs.queue.Queue` object) – The Queue from which messages are read.
- **`queue`** – The receipt handle associated with the message whose visibility timeout will be changed.
- **`visibility_timeout`** (*`int`*) – The new value of the message's visibility timeout in seconds.

`create_queue`(*queue_name*, *visibility_timeout=None*)
Create an SQS Queue.

Parameters

- **`queue_name`** (*`str or unicode`*) – The name of the new queue. Names are scoped to an account and need to be unique within that account. Calling this method on an existing queue name will not return an error from SQS unless the value for visibility_timeout is different than the value of the existing queue of that name. This is still an expensive operation, though, and not the preferred way to check for the existence of a queue. See the `boto.sqs.connection.SQSConnection.lookup()` method.
- **`visibility_timeout`** (*`int`*) – The default visibility timeout for all messages written in the queue. This can be overridden on a per-message.

Return type `boto.sqs.queue.Queue`

Returns The newly created queue.

`delete_message`(*queue*, *message*)
Delete a message from a queue.

Parameters

- **queue** (A *boto.sqs.queue.Queue* object) – The Queue from which messages are read.
- **message** (A *boto.sqs.message.Message* object) – The Message to be deleted

Return type bool

Returns True if successful, False otherwise.

delete_message_from_handle (*queue*, *receipt_handle*)
Delete a message from a queue, given a receipt handle.

Parameters

- **queue** (A *boto.sqs.queue.Queue* object) – The Queue from which messages are read.
- **receipt_handle** (*str*) – The receipt handle for the message

Return type bool

Returns True if successful, False otherwise.

delete_queue (*queue*, *force_deletion=False*)
Delete an SQS Queue.

Parameters

- **queue** (*A Queue object*) – The SQS queue to be deleted
- **force_deletion** (*Boolean*) – Normally, SQS will not delete a queue that contains messages. However, if the force_deletion argument is True, the queue will be deleted regardless of whether there are messages in the queue or not. USE WITH CAUTION. This will delete all messages in the queue as well.

Return type bool

Returns True if the command succeeded, False otherwise

get_all_queues (*prefix=''*)
Retrieves all queues.

Parameters **prefix** (*str*) – Optionally, only return queues that start with this value.

Return type *list*

Returns A list of *boto.sqs.queue.Queue* instances.

get_queue (*queue_name*)
Retrieves the queue with the given name, or `None` if no match was found.

Parameters **queue_name** (*str*) – The name of the queue to retrieve.

Return type *boto.sqs.queue.Queue* or `None`

Returns The requested queue, or `None` if no match was found.

get_queue_attributes (*queue*, *attribute='All'*)
Gets one or all attributes of a Queue

Parameters **queue** (*A Queue object*) – The SQS queue to be deleted

Return type *boto.sqs.attributes.Attributes*

Returns An Attributes object containing request value(s).

lookup (*queue_name*)
Retrieves the queue with the given name, or `None` if no match was found.

Parameters **queue_name** (*str*) – The name of the queue to retrieve.

Return type *boto.sqs.queue.Queue* or None

Returns The requested queue, or None if no match was found.

receive_message (*queue*, *number_messages=1*, *visibility_timeout=None*, *attributes=None*)
Read messages from an SQS Queue.

Parameters

- **queue** (*A Queue object*) – The Queue from which messages are read.
- **number_messages** (*int*) – The maximum number of messages to read (default=1)
- **visibility_timeout** (*int*) – The number of seconds the message should remain invisible to other queue readers (default=None which uses the Queues default)
- **attributes** (*str*) – The name of additional attribute to return with response or All if you want all attributes. The default is to return no additional attributes. Valid values:

  All|SenderId|SentTimestamp| ApproximateReceiveCount| ApproximateFirstReceive-Timestamp

Return type *list*

Returns A list of *boto.sqs.message.Message* objects.

remove_permission (*queue*, *label*)
Remove a permission from a queue.

Parameters

- **queue** (*boto.sqs.queue.Queue*) – The queue object
- **label** (*str or unicode*) – The unique label associated with the permission being removed.

Return type bool

Returns True if successful, False otherwise.

send_message (*queue*, *message_content*, *delay_seconds=None*)

send_message_batch (*queue*, *messages*)
Delivers up to 10 messages to a queue in a single request.

Parameters

- **queue** (A *boto.sqs.queue.Queue* object.) – The Queue to which the messages will be written.
- **messages** (*List of lists.*) – A list of lists or tuples. Each inner tuple represents a single message to be written and consists of and ID (string) that must be unique within the list of messages, the message body itself which can be a maximum of 64K in length, and an integer which represents the delay time (in seconds) for the message (0-900) before the message will be delivered to the queue.

set_queue_attribute (*queue*, *attribute*, *value*)

boto.sqs.jsonmessage

class boto.sqs.jsonmessage.**JSONMessage** (*queue=None*, *body=None*, *xml_attrs=None*)
Acts like a dictionary but encodes it's data as a Base64 encoded JSON payload.

decode (*value*)

encode (*value*)

boto.sqs.message

SQS Message

A Message represents the data stored in an SQS queue. The rules for what is allowed within an SQS Message are here:

> http://docs.amazonwebservices.com/AWSSimpleQueueService/2008-01-01/SQSDeveloperGuide/Query_QuerySendMessage.html

So, at it's simplest level a Message just needs to allow a developer to store bytes in it and get the bytes back out. However, to allow messages to have richer semantics, the Message class must support the following interfaces:

The constructor for the Message class must accept a keyword parameter "queue" which is an instance of a boto Queue object and represents the queue that the message will be stored in. The default value for this parameter is None.

The constructor for the Message class must accept a keyword parameter "body" which represents the content or body of the message. The format of this parameter will depend on the behavior of the particular Message subclass. For example, if the Message subclass provides dictionary-like behavior to the user the body passed to the constructor should be a dict-like object that can be used to populate the initial state of the message.

The Message class must provide an encode method that accepts a value of the same type as the body parameter of the constructor and returns a string of characters that are able to be stored in an SQS message body (see rules above).

The Message class must provide a decode method that accepts a string of characters that can be stored (and probably were stored!) in an SQS message and return an object of a type that is consistent with the "body" parameter accepted on the class constructor.

The Message class must provide a __len__ method that will return the size of the encoded message that would be stored in SQS based on the current state of the Message object.

The Message class must provide a get_body method that will return the body of the message in the same format accepted in the constructor of the class.

The Message class must provide a set_body method that accepts a message body in the same format accepted by the constructor of the class. This method should alter to the internal state of the Message object to reflect the state represented in the message body parameter.

The Message class must provide a get_body_encoded method that returns the current body of the message in the format in which it would be stored in SQS.

class `boto.sqs.message.`**`EncodedMHMessage`**(*queue=None*, *body=None*, *xml_attrs=None*)

The EncodedMHMessage class provides a message that provides RFC821-like headers like this:

HeaderName: HeaderValue

This variation encodes/decodes the body of the message in base64 automatically. The message instance can be treated like a mapping object, i.e. m['HeaderName'] would return 'HeaderValue'.

decode (*value*)

encode (*value*)

class `boto.sqs.message.`**`MHMessage`**(*queue=None*, *body=None*, *xml_attrs=None*)

The MHMessage class provides a message that provides RFC821-like headers like this:

HeaderName: HeaderValue

The encoding/decoding of this is handled automatically and after the message body has been read, the message instance can be treated like a mapping object, i.e. m['HeaderName'] would return 'HeaderValue'.

decode (*value*)

encode (*value*)

get (*key*, *default=None*)

has_key (*key*)

items ()

keys ()

update (*d*)

values ()

class `boto.sqs.message.`**Message** (*queue=None*, *body=''*)

The default Message class used for SQS queues. This class automatically encodes/decodes the message body using Base64 encoding to avoid any illegal characters in the message body. See:

http://developer.amazonwebservices.com/connect/thread.jspa?messageID=49680%EC%88%90

for details on why this is a good idea. The encode/decode is meant to be transparent to the end-user.

decode (*value*)

encode (*value*)

class `boto.sqs.message.`**RawMessage** (*queue=None*, *body=''*)

Base class for SQS messages. RawMessage does not encode the message in any way. Whatever you store in the body of the message is what will be written to SQS and whatever is returned from SQS is stored directly into the body of the message.

change_visibility (*visibility_timeout*)

decode (*value*)

Transform seralized byte array into any object.

delete ()

encode (*value*)

Transform body object into serialized byte array format.

endElement (*name*, *value*, *connection*)

get_body ()

get_body_encoded ()

This method is really a semi-private method used by the Queue.write method when writing the contents of the message to SQS. You probably shouldn't need to call this method in the normal course of events.

set_body (*body*)

Override the current body for this object, using decoded format.

startElement (*name*, *attrs*, *connection*)

boto.sqs.queue

Represents an SQS Queue

class `boto.sqs.queue.`**Queue** (*connection=None*, *url=None*, *message_class=<class boto.sqs.message.Message>*)

add_permission (*label*, *aws_account_id*, *action_name*)

Add a permission to a queue.

Parameters

- **label** (*str or unicode*) – A unique identification of the permission you are setting. Maximum of 80 characters `[0-9a-zA-Z_-]` Example, AliceSendMessage
- **principal_id** – The AWS account number of the principal who will be given permission. The principal must have an AWS account, but does not need to be signed up for Amazon SQS. For information about locating the AWS account identification.
- **action_name** (*str or unicode*) – The action. Valid choices are: *|SendMessage|ReceiveMessage|DeleteMessage| ChangeMessageVisibility|GetQueueAttributes

Return type bool

Returns True if successful, False otherwise.

clear (*page_size=10*, *vtimeout=10*)
Utility function to remove all messages from a queue

count (*page_size=10*, *vtimeout=10*)
Utility function to count the number of messages in a queue. Note: This function now calls GetQueueAttributes to obtain an 'approximate' count of the number of messages in a queue.

count_slow (*page_size=10*, *vtimeout=10*)
Deprecated. This is the old 'count' method that actually counts the messages by reading them all. This gives an accurate count but is very slow for queues with non-trivial number of messasges. Instead, use get_attribute('ApproximateNumberOfMessages') to take advantage of the new SQS capability. This is retained only for the unit tests.

delete ()
Delete the queue.

delete_message (*message*)
Delete a message from the queue.

Parameters **message** (*boto.sqs.message.Message*) – The *boto.sqs.message.Message* object to delete.

Return type bool

Returns True if successful, False otherwise

dump (*file_name*, *page_size=10*, *vtimeout=10*, *sep='\n'*)
Utility function to dump the messages in a queue to a file NOTE: Page size must be < 10 else SQS errors

endElement (*name*, *value*, *connection*)

get_attributes (*attributes='All'*)
Retrieves attributes about this queue object and returns them in an Attribute instance (subclass of a Dictionary).

Parameters **attributes** (*string*) – String containing one of: ApproximateNumberOfMessages, ApproximateNumberOfMessagesNotVisible, VisibilityTimeout, CreatedTimestamp, LastModifiedTimestamp, Policy

Return type Attribute object

Returns An Attribute object which is a mapping type holding the requested name/value pairs

get_messages (*num_messages=1*, *visibility_timeout=None*, *attributes=None*)
Get a variable number of messages.

Parameters

- **num_messages** (*int*) – The maximum number of messages to read from the queue.

- **visibility_timeout** (*int*) – The VisibilityTimeout for the messages read.
- **attributes** (*str*) – The name of additional attribute to return with response or All if you want all attributes. The default is to return no additional attributes. Valid values: All SenderId SentTimestamp ApproximateReceiveCount ApproximateFirstReceiveTimestamp

Return type *list*

Returns A list of *boto.sqs.message.Message* objects.

get_timeout()
Get the visibility timeout for the queue.

Return type int

Returns The number of seconds as an integer.

id

load(*file_name*, *sep='\n'*)
Utility function to load messages from a local filename to a queue

load_from_file(*fp*, *sep='\n'*)
Utility function to load messages from a file-like object to a queue

load_from_filename(*file_name*, *sep='\n'*)
Utility function to load messages from a local filename to a queue

load_from_s3(*bucket*, *prefix=None*)
Load messages previously saved to S3.

name

new_message(*body=''*)
Create new message of appropriate class.

Parameters **body** (*message body*) – The body of the newly created message (optional).

Return type *boto.sqs.message.Message*

Returns A new Message object

read(*visibility_timeout=None*)
Read a single message from the queue.

Parameters **visibility_timeout** (*int*) – The timeout for this message in seconds

Return type *boto.sqs.message.Message*

Returns A single message or None if queue is empty

remove_permission(*label*)
Remove a permission from a queue.

Parameters **label** (*str or unicode*) – The unique label associated with the permission being removed.

Return type bool

Returns True if successful, False otherwise.

save(*file_name*, *sep='\n'*)
Read all messages from the queue and persist them to local file. Messages are written to the file and the 'sep' string is written in between messages. Messages are deleted from the queue after being written to the file. Returns the number of messages saved.

save_to_file (*fp*, *sep='\n'*)
Read all messages from the queue and persist them to file-like object. Messages are written to the file and the 'sep' string is written in between messages. Messages are deleted from the queue after being written to the file. Returns the number of messages saved.

save_to_filename (*file_name*, *sep='\n'*)
Read all messages from the queue and persist them to local file. Messages are written to the file and the 'sep' string is written in between messages. Messages are deleted from the queue after being written to the file. Returns the number of messages saved.

save_to_s3 (*bucket*)
Read all messages from the queue and persist them to S3. Messages are stored in the S3 bucket using a naming scheme of:

```
<queue_id>/<message_id>
```

Messages are deleted from the queue after being saved to S3. Returns the number of messages saved.

set_attribute (*attribute*, *value*)
Set a new value for an attribute of the Queue.

Parameters

- **attribute** (*String*) – The name of the attribute you want to set. The only valid value at this time is: VisibilityTimeout
- **value** (*int*) – The new value for the attribute. For VisibilityTimeout the value must be an integer number of seconds from 0 to 86400.

Return type bool

Returns True if successful, otherwise False.

set_message_class (*message_class*)
Set the message class that should be used when instantiating messages read from the queue. By default, the class boto.sqs.message.Message is used but this can be overriden with any class that behaves like a message.

Parameters **message_class** (*Message-like class*) – The new Message class

set_timeout (*visibility_timeout*)
Set the visibility timeout for the queue.

Parameters **visibility_timeout** (*int*) – The desired timeout in seconds

startElement (*name*, *attrs*, *connection*)

write (*message*, *delay_seconds=None*)
Add a single message to the queue.

Parameters **message** (Message) – The message to be written to the queue

Return type *boto.sqs.message.Message*

Returns The *boto.sqs.message.Message* object that was written.

boto.sqs.regioninfo

class boto.sqs.regioninfo.**SQSRegionInfo** (*connection=None*, *name=None*, *endpoint=None*)

boto.sqs.batchresults

A set of results returned by SendMessageBatch.

class `boto.sqs.batchresults.`**`BatchResults`**(*parent*)

A container for the results of a send_message_batch request.

Variables

- **`results`** – A list of successful results. Each item in the list will be an instance of *`ResultEntry`*.
- **`errors`** – A list of unsuccessful results. Each item in the list will be an instance of *`ResultEntry`*.

`endElement`(*name*, *value*, *connection*)

`startElement`(*name*, *attrs*, *connection*)

class `boto.sqs.batchresults.`**`ResultEntry`**

The result (successful or unsuccessful) of a single message within a send_message_batch request.

In the case of a successful result, this dict-like object will contain the following items:

Variables

- ***`id`*** – A string containing the user-supplied ID of the message.
- ***`message_id`*** – A string containing the SQS ID of the new message.
- **`message_md5`** – A string containing the MD5 hash of the message body.

In the case of an error, this object will contain the following items:

Variables

- ***`id`*** – A string containing the user-supplied ID of the message.
- **`sender_fault`** – A boolean value.
- **`error_code`** – A string containing a short description of the error.
- **`error_message`** – A string containing a description of the error.

`endElement`(*name*, *value*, *connection*)

`startElement`(*name*, *attrs*, *connection*)

SNS

boto.sns

`boto.sns.`**`connect_to_region`**(*region_name*, ***kw_params*)

Given a valid region name, return a `boto.sns.connection.SNSConnection`.

Type str

Parameters **`region_name`** – The name of the region to connect to.

Return type `boto.sns.connection.SNSConnection` or `None`

Returns A connection to the given region, or None if an invalid region name is given `boto.sns.`**`get_region`**(*region_name*, ***kw_params*)

Find and return a `boto.regioninfo.RegionInfo` object given a region name.

Type str

Param The name of the region.

Return type `boto.regioninfo.RegionInfo`

Returns The RegionInfo object for the given region or None if an invalid region name is provided.

`boto.sns.`**`regions`**()

Get all available regions for the SNS service.

Return type *list*

Returns A list of `boto.regioninfo.RegionInfo` instances

class `boto.sns.`**`SNSConnection`**(*aws_access_key_id=None*, *aws_secret_access_key=None*, *is_secure=True*, *port=None*, *proxy=None*, *proxy_port=None*, *proxy_user=None*, *proxy_pass=None*, *debug=0*, *https_connection_factory=None*, *region=None*, *path='/'*, *security_token=None*)

`APIVersion` **= '2010-03-31'**

`DefaultRegionEndpoint` **= 'sns.us-east-1.amazonaws.com'**

`DefaultRegionName` **= 'us-east-1'**

`add_permission`(*topic*, *label*, *account_ids*, *actions*)

Adds a statement to a topic's access control policy, granting access for the specified AWS accounts to the specified actions.

Parameters

- **`topic`** (*`string`*) – The ARN of the topic.
- **`label`** (*`string`*) – A unique identifier for the new policy statement.
- **`account_ids`** (*`list of strings`*) – The AWS account ids of the users who will be give access to the specified actions.
- **`actions`** (*`list of strings`*) – The actions you want to allow for each of the specified principal(s).

`confirm_subscription`(*topic*, *token*, *authenticate_on_unsubscribe=False*)

Get properties of a Topic

Parameters

- **`topic`** (*`string`*) – The ARN of the new topic.
- **`token`** (*`string`*) – Short-lived token sent to and endpoint during the Subscribe operation.
- **`authenticate_on_unsubscribe`** (*`bool`*) – Optional parameter indicating that you wish to disable unauthenticated unsubscription of the subscription.

`create_topic`(*topic*)

Create a new Topic.

Parameters **`topic`** (*`string`*) – The name of the new topic.

`delete_topic`(*topic*)

Delete an existing topic

Parameters **`topic`** (*string*) – The ARN of the topic

`get_all_subscriptions`(*next_token=None*)
Get list of all subscriptions.

Parameters **`next_token`** (*string*) – Token returned by the previous call to this method.

`get_all_subscriptions_by_topic`(*topic*, *next_token=None*)
Get list of all subscriptions to a specific topic.

Parameters

- **`topic`** (*string*) – The ARN of the topic for which you wish to find subscriptions.
- **`next_token`** (*string*) – Token returned by the previous call to this method.

`get_all_topics`(*next_token=None*)

Parameters **`next_token`** (*string*) – Token returned by the previous call to this method.

`get_topic_attributes`(*topic*)
Get attributes of a Topic

Parameters **`topic`** (*string*) – The ARN of the topic.

`publish`(*topic*, *message*, *subject=None*)
Get properties of a Topic

Parameters

- **`topic`** (*string*) – The ARN of the new topic.
- **`message`** (*string*) – The message you want to send to the topic. Messages must be UTF-8 encoded strings and be at most 4KB in size.
- **`subject`** (*string*) – Optional parameter to be used as the "Subject" line of the email notifications.

`remove_permission`(*topic*, *label*)
Removes a statement from a topic's access control policy.

Parameters

- **`topic`** (*string*) – The ARN of the topic.
- **`label`** (*string*) – A unique identifier for the policy statement to be removed.

`set_topic_attributes`(*topic*, *attr_name*, *attr_value*)
Get attributes of a Topic

Parameters

- **`topic`** (*string*) – The ARN of the topic.
- **`attr_name`** (*string*) – The name of the attribute you want to set. Only a subset of the topic's attributes are mutable. Valid values: Policy | DisplayName
- **`attr_value`** (*string*) – The new value for the attribute.

`subscribe`(*topic*, *protocol*, *endpoint*)
Subscribe to a Topic.

Parameters

- **`topic`** (*string*) – The name of the new topic.
- **`protocol`** (*string*) – The protocol used to communicate with the subscriber. Current choices are: email|email-json|http|https|sqs

- **endpoint** (*string*) – The location of the endpoint for the subscriber. * For email, this would be a valid email address * For email-json, this would be a valid email address * For http, this would be a URL beginning with http * For https, this would be a URL beginning with https * For sqs, this would be the ARN of an SQS Queue

subscribe_sqs_queue (*topic*, *queue*)
Subscribe an SQS queue to a topic.

This is convenience method that handles most of the complexity involved in using ans SQS queue as an endpoint for an SNS topic. To achieve this the following operations are performed:

- The correct ARN is constructed for the SQS queue and that ARN is then subscribed to the topic.
- A JSON policy document is contructed that grants permission to the SNS topic to send messages to the SQS queue.
- This JSON policy is then associated with the SQS queue using the queue's set_attribute method. If the queue already has a policy associated with it, this process will add a Statement to that policy. If no policy exists, a new policy will be created.

Parameters

- **topic** (*string*) – The name of the new topic.
- **queue** (*A boto Queue object*) – The queue you wish to subscribe to the SNS Topic.

unsubscribe (*subscription*)
Allows endpoint owner to delete subscription. Confirmation message will be delivered.

Parameters **subscription** (*string*) – The ARN of the subscription to be deleted.

Simple Email Service Tutorial

This tutorial focuses on the boto interface to AWS' Simple Email Service (SES). This tutorial assumes that you have boto already downloaded and installed.

Creating a Connection

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

```
>>> import boto
>>> conn = boto.connect_ses(
        aws_access_key_id='<YOUR_AWS_KEY_ID>',
        aws_secret_access_key='<YOUR_AWS_SECRET_KEY>')
>>> conn
SESConnection:email.us-east-1.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. More details on configuration can be fond in *Boto Config*.

The *boto.connect_ses()* functions returns a *boto.ses.connection.SESConnection* instance, which is a the boto API for working with SES.

Notes on Sending

It is important to keep in mind that while emails appear to come "from" the address that you specify via Reply-To, the sending is done through Amazon. Some clients do pick up on this disparity, and leave a note on emails.

Verifying a Sender Email Address

Before you can send email "from" an address, you must prove that you have access to the account. When you send a validation request, an email is sent to the address with a link in it. Clicking on the link validates the address and adds it to your SES account. Here's how to send the validation email:

```
>>> conn.verify_email_address('some@address.com')
{
    'VerifyEmailAddressResponse': {
        'ResponseMetadata': {
            'RequestId': '4a974fd5-56c2-11e1-ad4c-c1f08c91d554'
        }
    }
}
```

After a short amount of time, you'll find an email with the validation link inside. Click it, and this address may be used to send emails.

Listing Verified Addresses

If you'd like to list the addresses that are currently verified on your SES account, use *list_verified_email_addresses*:

```
>>> conn.list_verified_email_addresses()
{
    'ListVerifiedEmailAddressesResponse': {
        'ListVerifiedEmailAddressesResult': {
            'VerifiedEmailAddresses': [
                'some@address.com',
                'another@address.com'
            ]
        },
        'ResponseMetadata': {
            'RequestId': '2ab45c18-56c3-11e1-be66-ffd2a4549d70'
        }
    }
}
```

Deleting a Verified Address

In the event that you'd like to remove an email address from your account, use *delete_verified_email_address*:

```
>>> conn.delete_verified_email_address('another@address.com')
```

Sending an Email

Sending an email is done via *send_email*:

```
>>> conn.send_email(
        'some@address.com',
        'Your subject',
        'Body here',
        ['recipient-address-1@gmail.com'])
{
    'SendEmailResponse': {
        'ResponseMetadata': {
            'RequestId': '4743c2b7-56c3-11e1-bccd-c99bd68002fd'
        },
        'SendEmailResult': {
            'MessageId': '000001357a177192-7b894025-147a-4705-8455-7c880b0c8270-000000
↪'
        }
    }
}
```

If you're wanting to send a multipart MIME email, see the reference for *send_raw_email*, which is a bit more of a low-level alternative.

Checking your Send Quota

Staying within your quota is critical, since the upper limit is a hard cap. Once you have hit your quota, no further email may be sent until enough time elapses to where your 24 hour email count (rolling continuously) is within acceptable ranges. Use *get_send_quota*:

```
>>> conn.get_send_quota()
{
    'GetSendQuotaResponse': {
        'GetSendQuotaResult': {
            'Max24HourSend': '100000.0',
            'SentLast24Hours': '181.0',
            'MaxSendRate': '28.0'
        },
        'ResponseMetadata': {
            'RequestId': u'8a629245-56c4-11e1-9c53-9d5f4d2cc8d3'
        }
    }
}
```

Checking your Send Statistics

In order to fight spammers and ensure quality mail is being sent from SES, Amazon tracks bounces, rejections, and complaints. This is done via *get_send_statistics*. Please be warned that the output is extremely verbose, to the point where we'll just show a short excerpt here:

```
>>> conn.get_send_statistics()
{
    'GetSendStatisticsResponse': {
        'GetSendStatisticsResult': {
            'SendDataPoints': [
                {
                    'Complaints': '0',
                    'Timestamp': '2012-02-13T05:02:00Z',
                    'DeliveryAttempts': '8',
                    'Bounces': '0',
                    'Rejects': '0'
                },
                {
                    'Complaints': '0',
                    'Timestamp': '2012-02-13T05:17:00Z',
                    'DeliveryAttempts': '12',
                    'Bounces': '0',
                    'Rejects': '0'
                }
            ]
        }
    }
}
```

SES

boto.ses

`boto.ses.`**`connect_to_region`**`(`*region_name*, ***kw_params*`)`

Given a valid region name, return a `boto.sns.connection.SESConnection`.

Type str

Parameters **`region_name`** – The name of the region to connect to.

Return type `boto.sns.connection.SESConnection` or `None`

Returns A connection to the given region, or None if an invalid region name is given

`boto.ses.`**`get_region`**`(`*region_name*, ***kw_params*`)`

Find and return a `boto.regioninfo.RegionInfo` object given a region name.

Type str

Param The name of the region.

Return type `boto.regioninfo.RegionInfo`

Returns The RegionInfo object for the given region or None if an invalid region name is provided.

`boto.ses.`**`regions`**`()`

Get all available regions for the SES service.

Return type *list*

Returns A list of `boto.regioninfo.RegionInfo` instances

boto.ses.connection

class `boto.ses.connection.`**`SESConnection`**(*aws_access_key_id=None, aws_secret_access_key=None, is_secure=True, port=None, proxy=None, proxy_port=None, proxy_user=None, proxy_pass=None, debug=0, https_connection_factory=None, region=None, path='/'*)

`APIVersion` = **'2010-12-01'**

`DefaultRegionEndpoint` = **'email.us-east-1.amazonaws.com'**

`DefaultRegionName` = **'us-east-1'**

`ResponseError`
alias of `BotoServerError`

`delete_verified_email_address`(*email_address*)
Deletes the specified email address from the list of verified addresses.

Parameters **`email_address`** – The email address to be removed from the list of verified addreses.

Return type *dict*

Returns A DeleteVerifiedEmailAddressResponse structure. Note that keys must be unicode strings.

`get_send_quota`()
Fetches the user's current activity limits.

Return type *dict*

Returns A GetSendQuotaResponse structure. Note that keys must be unicode strings.

`get_send_statistics`()
Fetches the user's sending statistics. The result is a list of data points, representing the last two weeks of sending activity.

Each data point in the list contains statistics for a 15-minute interval.

Return type *dict*

Returns A GetSendStatisticsResponse structure. Note that keys must be unicode strings.

`list_verified_email_addresses`()
Fetch a list of the email addresses that have been verified.

Return type *dict*

Returns A ListVerifiedEmailAddressesResponse structure. Note that keys must be unicode strings.

`send_email`(*source, subject, body, to_addresses, cc_addresses=None, bcc_addresses=None, format='text', reply_addresses=None, return_path=None, text_body=None, html_body=None*)
Composes an email message based on input data, and then immediately queues the message for sending.

Parameters

- **`source`** (*`string`*) – The sender's email address.
- **`subject`** (*`string`*) – The subject of the message: A short summary of the content, which will appear in the recipient's inbox.

- **body** (*string*) – The message body.
- **to_addresses** (*list of strings or string*) – The To: field(s) of the message.
- **cc_addresses** (*list of strings or string*) – The CC: field(s) of the message.
- **bcc_addresses** (*list of strings or string*) – The BCC: field(s) of the message.
- **format** (*string*) – The format of the message's body, must be either "text" or "html".
- **reply_addresses** (*list of strings or string*) – The reply-to email address(es) for the message. If the recipient replies to the message, each reply-to address will receive the reply.
- **return_path** (*string*) – The email address to which bounce notifications are to be forwarded. If the message cannot be delivered to the recipient, then an error message will be returned from the recipient's ISP; this message will then be forwarded to the email address specified by the ReturnPath parameter.
- **text_body** (*string*) – The text body to send with this email.
- **html_body** (*string*) – The html body to send with this email.

send_raw_email(*raw_message*, *source=None*, *destinations=None*)

Sends an email message, with header and content specified by the client. The SendRawEmail action is useful for sending multipart MIME emails, with attachments or inline content. The raw text of the message must comply with Internet email standards; otherwise, the message cannot be sent.

Parameters

- **source** (*string*) – The sender's email address. Amazon's docs say:

  If you specify the Source parameter, then bounce notifications and complaints will be sent to this email address. This takes precedence over any Return-Path header that you might include in the raw text of the message.

- **raw_message** (*string*) – The raw text of the message. The client is responsible for ensuring the following:
  - Message must contain a header and a body, separated by a blank line.
  - All required header fields must be present.
  - Each part of a multipart MIME message must be formatted properly.
  - MIME content types must be among those supported by Amazon SES. Refer to the Amazon SES Developer Guide for more details.
  - Content must be base64-encoded, if MIME requires it.
- **destinations** (*list of strings or string*) – A list of destinations for the message.

verify_email_address(*email_address*)

Verifies an email address. This action causes a confirmation email message to be sent to the specified address.

Parameters **email_address** – The email address to be verified.

Return type *dict*

Returns A VerifyEmailAddressResponse structure. Note that keys must be unicode strings.

CloudWatch

First, make sure you have something to monitor. You can either create a LoadBalancer or enable monitoring on an existing EC2 instance. To enable monitoring, you can either call the monitor_instance method on the EC2Connection object or call the monitor method on the Instance object.

It takes a while for the monitoring data to start accumulating but once it does, you can do this:

```
>>> import boto
>>> c = boto.connect_cloudwatch()
>>> metrics = c.list_metrics()
>>> metrics
[Metric:NetworkIn,
 Metric:NetworkOut,
 Metric:NetworkOut(InstanceType,m1.small),
 Metric:NetworkIn(InstanceId,i-e573e68c),
 Metric:CPUUtilization(InstanceId,i-e573e68c),
 Metric:DiskWriteBytes(InstanceType,m1.small),
 Metric:DiskWriteBytes(ImageId,ami-a1ffb63),
 Metric:NetworkOut(ImageId,ami-a1ffb63),
 Metric:DiskWriteOps(InstanceType,m1.small),
 Metric:DiskReadBytes(InstanceType,m1.small),
 Metric:DiskReadOps(ImageId,ami-a1ffb63),
 Metric:CPUUtilization(InstanceType,m1.small),
 Metric:NetworkIn(ImageId,ami-a1ffb63),
 Metric:DiskReadOps(InstanceType,m1.small),
 Metric:DiskReadBytes,
 Metric:CPUUtilization,
 Metric:DiskWriteBytes(InstanceId,i-e573e68c),
 Metric:DiskWriteOps(InstanceId,i-e573e68c),
 Metric:DiskWriteOps,
 Metric:DiskReadOps,
 Metric:CPUUtilization(ImageId,ami-a1ffb63),
 Metric:DiskReadOps(InstanceId,i-e573e68c),
 Metric:NetworkOut(InstanceId,i-e573e68c),
 Metric:DiskReadBytes(ImageId,ami-a1ffb63),
 Metric:DiskReadBytes(InstanceId,i-e573e68c),
 Metric:DiskWriteBytes,
 Metric:NetworkIn(InstanceType,m1.small),
 Metric:DiskWriteOps(ImageId,ami-a1ffb63)]
```

The list_metrics call will return a list of all of the available metrics that you can query against. Each entry in the list is a Metric object. As you can see from the list above, some of the metrics are generic metrics and some have Dimensions associated with them (e.g. InstanceType=m1.small). The Dimension can be used to refine your query. So, for example, I could query the metric Metric:CPUUtilization which would create the desired statistic by aggregating cpu utilization data across all sources of information available or I could refine that by querying the metric Metric:CPUUtilization(InstanceId,i-e573e68c) which would use only the data associated with the instance identified by the instance ID i-e573e68c.

Because for this example, I'm only monitoring a single instance, the set of metrics available to me are fairly limited. If I was monitoring many instances, using many different instance types and AMI's and also several load balancers, the list of available metrics would grow considerably.

Once you have the list of available metrics, you can actually query the CloudWatch system for that metric. Let's choose the CPU utilization metric for our instance.:

```
>>> m = metrics[5]
>>> m
```

```
Metric:CPUUtilization(InstanceId,i-e573e68c)
```

The Metric object has a query method that lets us actually perform the query against the collected data in CloudWatch. To call that, we need a start time and end time to control the time span of data that we are interested in. For this example, let's say we want the data for the previous hour:

```
>>> import datetime
>>> end = datetime.datetime.now()
>>> start = end - datetime.timedelta(hours=1)
```

We also need to supply the Statistic that we want reported and the Units to use for the results. The Statistic can be one of these values:

```
['Minimum', 'Maximum', 'Sum', 'Average', 'SampleCount']
```

And Units must be one of the following:

```
['Seconds', 'Percent', 'Bytes', 'Bits', 'Count',
'Bytes/Second', 'Bits/Second', 'Count/Second']
```

The query method also takes an optional parameter, period. This parameter controls the granularity (in seconds) of the data returned. The smallest period is 60 seconds and the value must be a multiple of 60 seconds. So, let's ask for the average as a percent:

```
>>> datapoints = m.query(start, end, 'Average', 'Percent')
>>> len(datapoints)
60
```

Our period was 60 seconds and our duration was one hour so we should get 60 data points back and we can see that we did. Each element in the datapoints list is a DataPoint object which is a simple subclass of a Python dict object. Each Datapoint object contains all of the information available about that particular data point.:

```
>>> d = datapoints[0]
>>> d
{u'Average': 0.0,
 u'SampleCount': 1.0,
 u'Timestamp': u'2009-05-21T19:55:00Z',
 u'Unit': u'Percent'}
```

My server obviously isn't very busy right now!

CloudWatch Reference

boto.ec2.cloudwatch

This module provides an interface to the Elastic Compute Cloud (EC2) CloudWatch service from AWS.

class `boto.ec2.cloudwatch.`**`CloudWatchConnection`**(*aws_access_key_id=None, aws_secret_access_key=None, is_secure=True, port=None, proxy=None, proxy_port=None, proxy_user=None, proxy_pass=None, debug=0, https_connection_factory=None, region=None, path='/'*)

Init method to create a new connection to EC2 Monitoring Service.

B{Note:} The host argument is overridden by the host specified in the boto configuration file.

APIVersion = '2010-08-01'

DefaultRegionEndpoint = 'monitoring.amazonaws.com'

DefaultRegionName = 'us-east-1'

build_dimension_param (*dimension*, *params*)

build_list_params (*params*, *items*, *label*)

build_put_params (*params*, *name*, *value=None*, *timestamp=None*, *unit=None*, *dimensions=None*, *statistics=None*)

create_alarm (*alarm*)

Creates or updates an alarm and associates it with the specified Amazon CloudWatch metric. Optionally, this operation can associate one or more Amazon Simple Notification Service resources with the alarm.

When this operation creates an alarm, the alarm state is immediately set to INSUFFICIENT_DATA. The alarm is evaluated and its StateValue is set appropriately. Any actions associated with the StateValue is then executed.

When updating an existing alarm, its StateValue is left unchanged.

Parameters alarm (*boto.ec2.cloudwatch.alarm.MetricAlarm*) – MetricAlarm object.

delete_alarms (*alarms*)

Deletes all specified alarms. In the event of an error, no alarms are deleted.

Parameters alarms (list) – List of alarm names.

describe_alarm_history (*alarm_name=None*, *start_date=None*, *end_date=None*, *max_records=None*, *history_item_type=None*, *next_token=None*)

Retrieves history for the specified alarm. Filter alarms by date range or item type. If an alarm name is not specified, Amazon CloudWatch returns histories for all of the owner's alarms.

Amazon CloudWatch retains the history of deleted alarms for a period of six weeks. If an alarm has been deleted, its history can still be queried.

Parameters

- **alarm_name** (*string*) – The name of the alarm.
- **start_date** (*datetime*) – The starting date to retrieve alarm history.
- **end_date** (*datetime*) – The starting date to retrieve alarm history.
- **history_item_type** (*string*) – The type of alarm histories to retreive (ConfigurationUpdate | StateUpdate | Action)
- **max_records** (*int*) – The maximum number of alarm descriptions to retrieve.
- **next_token** (*string*) – The token returned by a previous call to indicate that there is more data.

:rtype list

describe_alarms (*action_prefix=None*, *alarm_name_prefix=None*, *alarm_names=None*, *max_records=None*, *state_value=None*, *next_token=None*)

Retrieves alarms with the specified names. If no name is specified, all alarms for the user are returned. Alarms can be retrieved by using only a prefix for the alarm name, the alarm state, or a prefix for any action.

Parameters

- **action_name** – The action name prefix.
- **alarm_name_prefix** (*string*) – The alarm name prefix. AlarmNames cannot be specified if this parameter is specified.
- **alarm_names** (`list`) – A list of alarm names to retrieve information for.
- **max_records** (*int*) – The maximum number of alarm descriptions to retrieve.
- **state_value** (*string*) – The state value to be used in matching alarms.
- **next_token** (*string*) – The token returned by a previous call to indicate that there is more data.

:rtype list

describe_alarms_for_metric(*metric_name*, *namespace*, *period=None*, *statistic=None*, *dimensions=None*, *unit=None*)

Retrieves all alarms for a single metric. Specify a statistic, period, or unit to filter the set of alarms further.

Parameters

- **metric_name** (*string*) – The name of the metric
- **namespace** (*string*) – The namespace of the metric.
- **period** (*int*) – The period in seconds over which the statistic is applied.
- **statistic** (*string*) – The statistic for the metric.
- **dimension_filters** – A dictionary containing name/value pairs that will be used to filter the results. The key in the dictionary is the name of a Dimension. The value in the dictionary is either a scalar value of that Dimension name that you want to filter on, a list of values to filter on or None if you want all metrics with that Dimension name.

:rtype list

disable_alarm_actions(*alarm_names*)

Disables actions for the specified alarms.

Parameters alarms (`list`) – List of alarm names.

enable_alarm_actions(*alarm_names*)

Enables actions for the specified alarms.

Parameters alarms (`list`) – List of alarm names.

get_metric_statistics(*period*, *start_time*, *end_time*, *metric_name*, *namespace*, *statistics*, *dimensions=None*, *unit=None*)

Get time-series data for one or more statistics of a given metric.

Parameters

- **period** (`integer`) – The granularity, in seconds, of the returned datapoints. Period must be at least 60 seconds and must be a multiple of 60. The default value is 60.
- **start_time** (*datetime*) – The time stamp to use for determining the first datapoint to return. The value specified is inclusive; results include datapoints with the time stamp specified.
- **end_time** (*datetime*) – The time stamp to use for determining the last datapoint to return. The value specified is exclusive; results will include datapoints up to the time stamp specified.
- **metric_name** (*string*) – The metric name.
- **namespace** (*string*) – The metric's namespace.

- **statistics** (list) – A list of statistics names Valid values: Average | Sum | SampleCount | Maximum | Minimum
- **dimensions** (dict) – A dictionary of dimension key/values where the key is the dimension name and the value is either a scalar value or an iterator of values to be associated with that dimension.

Return type *list*

list_metrics (*next_token=None*, *dimensions=None*, *metric_name=None*, *namespace=None*)
Returns a list of the valid metrics for which there is recorded data available.

Parameters

- **next_token** (*str*) – A maximum of 500 metrics will be returned at one time. If more results are available, the ResultSet returned will contain a non-Null next_token attribute. Passing that token as a parameter to list_metrics will retrieve the next page of metrics.
- **dimension_filters** – A dictionary containing name/value pairs that will be used to filter the results. The key in the dictionary is the name of a Dimension. The value in the dictionary is either a scalar value of that Dimension name that you want to filter on, a list of values to filter on or None if you want all metrics with that Dimension name.
- **metric_name** (*str*) – The name of the Metric to filter against. If None, all Metric names will be returned.
- **namespace** (*str*) – A Metric namespace to filter against (e.g. AWS/EC2). If None, Metrics from all namespaces will be returned.

put_metric_alarm (*alarm*)
Creates or updates an alarm and associates it with the specified Amazon CloudWatch metric. Optionally, this operation can associate one or more Amazon Simple Notification Service resources with the alarm.

When this operation creates an alarm, the alarm state is immediately set to INSUFFICIENT_DATA. The alarm is evaluated and its StateValue is set appropriately. Any actions associated with the StateValue is then executed.

When updating an existing alarm, its StateValue is left unchanged.

Parameters alarm (*boto.ec2.cloudwatch.alarm.MetricAlarm*) – MetricAlarm object.

put_metric_data (*namespace*, *name*, *value=None*, *timestamp=None*, *unit=None*, *dimensions=None*, *statistics=None*)
Publishes metric data points to Amazon CloudWatch. Amazon Cloudwatch associates the data points with the specified metric. If the specified metric does not exist, Amazon CloudWatch creates the metric. If a list is specified for some, but not all, of the arguments, the remaining arguments are repeated a corresponding number of times.

Parameters

- **namespace** (*str*) – The namespace of the metric.
- **name** (*str or list*) – The name of the metric.
- **value** (*float or list*) – The value for the metric.
- **timestamp** (*datetime or list*) – The time stamp used for the metric. If not specified, the default value is set to the time the metric data was received.
- **unit** (*string or list*) – The unit of the metric. Valid Values: Seconds | Microseconds | Milliseconds | Bytes | Kilobytes | Megabytes | Gigabytes | Terabytes | Bits | Kilobits | Megabits | Gigabits | Terabits | Percent | Count | Bytes/Second | Kilobytes/Second |

Megabytes/Second | Gigabytes/Second | Terabytes/Second | Bits/Second | Kilobits/Second | Megabits/Second | Gigabits/Second | Terabits/Second | Count/Second | None

- **dimensions** (dict) – Add extra name value pairs to associate with the metric, i.e.: {'name1': value1, 'name2': (value2, value3)}
- **statistics** (*dict or list*) – Use a statistic set instead of a value, for example:

```
{'maximum': 30, 'minimum': 1, 'samplecount': 100, 'sum': 10000}
```

set_alarm_state(*alarm_name*, *state_reason*, *state_value*, *state_reason_data=None*)
Temporarily sets the state of an alarm. When the updated StateValue differs from the previous value, the action configured for the appropriate state is invoked. This is not a permanent change. The next periodic alarm check (in about a minute) will set the alarm to its actual state.

Parameters

- **alarm_name** (*string*) – Descriptive name for alarm.
- **state_reason** (*string*) – Human readable reason.
- **state_value** (*string*) – OK | ALARM | INSUFFICIENT_DATA
- **state_reason_data** (*string*) – Reason string (will be jsonified).

update_alarm(*alarm*)
Creates or updates an alarm and associates it with the specified Amazon CloudWatch metric. Optionally, this operation can associate one or more Amazon Simple Notification Service resources with the alarm.

When this operation creates an alarm, the alarm state is immediately set to INSUFFICIENT_DATA. The alarm is evaluated and its StateValue is set appropriately. Any actions associated with the StateValue is then executed.

When updating an existing alarm, its StateValue is left unchanged.

Parameters **alarm** (*boto.ec2.cloudwatch.alarm.MetricAlarm*) – MetricAlarm object.

boto.ec2.cloudwatch.**connect_to_region**(*region_name*, ***kw_params*)
Given a valid region name, return a *boto.ec2.cloudwatch.CloudWatchConnection*.

Parameters **region_name** (*str*) – The name of the region to connect to.

Return type boto.ec2.CloudWatchConnection or None

Returns A connection to the given region, or None if an invalid region name is given

boto.ec2.cloudwatch.**regions**()
Get all available regions for the CloudWatch service.

Return type *list*

Returns A list of boto.RegionInfo instances

boto.ec2.cloudwatch.datapoint

class boto.ec2.cloudwatch.datapoint.**Datapoint**(*connection=None*)

endElement(*name*, *value*, *connection*)

startElement(*name*, *attrs*, *connection*)

boto.ec2.cloudwatch.metric

class `boto.ec2.cloudwatch.metric.`**`Metric`**(*connection=None*)

> **`Statistics`** = **['Minimum', 'Maximum', 'Sum', 'Average', 'SampleCount']**
>
> **`Units`** = **['Seconds', 'Microseconds', 'Milliseconds', 'Bytes', 'Kilobytes', 'Megabytes', 'Gigabytes', 'Terabytes', 'Bits', 'K**
>
> **`create_alarm`**(*name*, *comparison*, *threshold*, *period*, *evaluation_periods*, *statistic*, *enabled=True*, *description=None*, *dimensions=None*, *alarm_actions=None*, *ok_actions=None*, *insufficient_data_actions=None*, *unit=None*)
>
> **`describe_alarms`**(*period=None*, *statistic=None*, *dimensions=None*, *unit=None*)
>
> **`endElement`**(*name*, *value*, *connection*)
>
> **`query`**(*start_time*, *end_time*, *statistics*, *unit=None*, *period=60*)
>
> **`startElement`**(*name*, *attrs*, *connection*)

route53

boto.route53.connection

class `boto.route53.connection.`**`Route53Connection`**(*aws_access_key_id=None*, *aws_secret_access_key=None*, *port=None*, *proxy=None*, *proxy_port=None*, *host='route53.amazonaws.com'*, *debug=0*)

> **`DefaultHost`** = **'route53.amazonaws.com'**
> The default Route53 API endpoint to connect to.
>
> **`Version`** = **'2011-05-05'**
> Route53 API version.
>
> **`XMLNameSpace`** = **'https://route53.amazonaws.com/doc/2011-05-05/'**
> XML schema for this Route53 API version.
>
> **`change_rrsets`**(*hosted_zone_id*, *xml_body*)
> Create or change the authoritative DNS information for this Hosted Zone. Returns a Python data structure with information about the set of changes, including the Change ID.
>
> **Parameters**
>
> - **`hosted_zone_id`** (`str`) – The unique identifier for the Hosted Zone
> - **`xml_body`** (`str`) – The list of changes to be made, defined in the XML schema defined by the Route53 service.
>
> **`create_hosted_zone`**(*domain_name*, *caller_ref=None*, *comment=''*)
> Create a new Hosted Zone. Returns a Python data structure with information about the newly created Hosted Zone.
>
> **Parameters**
>
> - **`domain_name`** (`str`) – The name of the domain. This should be a fully-specified domain, and should end with a final period as the last label indication. If you omit the final

period, Amazon Route 53 assumes the domain is relative to the root. This is the name you have registered with your DNS registrar. It is also the name you will delegate from your registrar to the Amazon Route 53 delegation servers returned in response to this request.A list of strings with the image IDs wanted.

- **caller_ref** (*str*) – A unique string that identifies the request and that allows failed CreateHostedZone requests to be retried without the risk of executing the operation twice. If you don't provide a value for this, boto will generate a Type 4 UUID and use that.
- **comment** (*str*) – Any comments you want to include about the hosted zone.

delete_hosted_zone(*hosted_zone_id*)

get_all_hosted_zones(*start_marker=None*, *zone_list=None*)
Returns a Python data structure with information about all Hosted Zones defined for the AWS account.

Parameters

- **start_marker** (*int*) – start marker to pass when fetching additional results after a truncated list
- **zone_list** (*list*) – a HostedZones list to prepend to results

get_all_rrsets(*hosted_zone_id*, *type=None*, *name=None*, *identifier=None*, *maxitems=None*)
Retrieve the Resource Record Sets defined for this Hosted Zone. Returns the raw XML data returned by the Route53 call.

Parameters

- **hosted_zone_id** (*str*) – The unique identifier for the Hosted Zone
- **type** (*str*) – The type of resource record set to begin the record listing from. Valid choices are:
  - A
  - AAAA
  - CNAME
  - MX
  - NS
  - PTR
  - SOA
  - SPF
  - SRV
  - TXT

  Valid values for weighted resource record sets:
  - A
  - AAAA
  - CNAME
  - TXT

  Valid values for Zone Apex Aliases:
  - A

  – AAAA

- **name** (*str*) – The first name in the lexicographic ordering of domain names to be retrieved
- **identifier** (*str*) – In a hosted zone that includes weighted resource record sets (multiple resource record sets with the same DNS name and type that are differentiated only by SetIdentifier), if results were truncated for a given DNS name and type, the value of SetIdentifier for the next resource record set that has the current DNS name and type
- **maxitems** (*int*) – The maximum number of records

get_change (*change_id*)
Get information about a proposed set of changes, as submitted by the change_rrsets method. Returns a Python data structure with status information about the changes.

Parameters change_id (*str*) – The unique identifier for the set of changes. This ID is returned in the response to the change_rrsets method.

get_hosted_zone (*hosted_zone_id*)
Get detailed information about a particular Hosted Zone.

Parameters hosted_zone_id (*str*) – The unique identifier for the Hosted Zone

make_request (*action, path, headers=None, data='', params=None*)

boto.route53.hostedzone

class boto.route53.hostedzone.**HostedZone** (*id=None, name=None, owner=None, version=None, caller_reference=None, config=None*)

endElement (*name, value, connection*)

startElement (*name, attrs, connection*)

boto.route53.exception

exception boto.route53.exception.**DNSServerError** (*status, reason, body=None, *args*)

An Introduction to boto's VPC interface

This tutorial is based on the examples in the Amazon Virtual Private Cloud Getting Started Guide (http://docs.amazonwebservices.com/AmazonVPC/latest/GettingStartedGuide/). In each example, it tries to show the boto request that correspond to the AWS command line tools.

Creating a VPC connection

First, we need to create a new VPC connection:

```
>>> from boto.vpc import VPCConnection
>>> c = VPCConnection()
```

To create a VPC

Now that we have a VPC connection, we can create our first VPC.

```
>>> vpc = c.create_vpc('10.0.0.0/24')
>>> vpc
VPC:vpc-6b1fe402
>>> vpc.id
u'vpc-6b1fe402'
>>> vpc.state
u'pending'
>>> vpc.cidr_block
u'10.0.0.0/24'
>>> vpc.dhcp_options_id
u'default'
>>>
```

To create a subnet

The next step is to create a subnet to associate with your VPC.

```
>>> subnet = c.create_subnet(vpc.id, '10.0.0.0/25')
>>> subnet.id
u'subnet-6a1fe403'
>>> subnet.state
u'pending'
>>> subnet.cidr_block
u'10.0.0.0/25'
>>> subnet.available_ip_address_count
123
>>> subnet.availability_zone
u'us-east-1b'
>>>
```

To create a customer gateway

Next, we create a customer gateway.

```
>>> cg = c.create_customer_gateway('ipsec.1', '12.1.2.3', 65534)
>>> cg.id
u'cgw-b6a247df'
>>> cg.type
u'ipsec.1'
>>> cg.state
u'available'
>>> cg.ip_address
u'12.1.2.3'
>>> cg.bgp_asn
u'65534'
>>>
```

To create a VPN gateway

```
>>> vg = c.create_vpn_gateway('ipsec.1')
>>> vg.id
u'vgw-44ad482d'
>>> vg.type
u'ipsec.1'
>>> vg.state
u'pending'
>>> vg.availability_zone
u'us-east-1b'
>>>
```

Attaching a VPN Gateway to a VPC

```
>>> vg.attach(vpc.id)
>>>
```

VPC

boto.vpc

Represents a connection to the EC2 service.

class `boto.vpc.`**`VPCConnection`**(*aws_access_key_id=None*, *aws_secret_access_key=None*, *is_secure=True*, *host=None*, *port=None*, *proxy=None*, *proxy_port=None*, *proxy_user=None*, *proxy_pass=None*, *debug=0*, *https_connection_factory=None*, *region=None*, *path='/'*, *api_version=None*, *security_token=None*)

Init method to create a new connection to EC2.

`associate_dhcp_options`(*dhcp_options_id*, *vpc_id*)

Associate a set of Dhcp Options with a VPC.

Parameters

- **`dhcp_options_id`** (`str`) – The ID of the Dhcp Options
- **`vpc_id`** (`str`) – The ID of the VPC.

Return type bool

Returns True if successful

`associate_route_table`(*route_table_id*, *subnet_id*)

Associates a route table with a specific subnet.

Parameters

- **`route_table_id`** (`str`) – The ID of the route table to associate.
- **`subnet_id`** (`str`) – The ID of the subnet to associate with.

Return type str

Returns The ID of the association created

attach_internet_gateway(*internet_gateway_id*, *vpc_id*)
Attach an internet gateway to a specific VPC.

Parameters

- **internet_gateway_id** (*str*) – The ID of the internet gateway to delete.
- **vpc_id** (*str*) – The ID of the VPC to attach to.

Return type Bool

Returns True if successful

attach_vpn_gateway(*vpn_gateway_id*, *vpc_id*)
Attaches a VPN gateway to a VPC.

Parameters

- **vpn_gateway_id** (*str*) – The ID of the vpn_gateway to attach
- **vpc_id** (*str*) – The ID of the VPC you want to attach the gateway to.

Return type An attachment

Returns a *boto.vpc.vpngateway.Attachment*

create_customer_gateway(*type*, *ip_address*, *bgp_asn*)
Create a new Customer Gateway

Parameters

- **type** (*str*) – Type of VPN Connection. Only valid valid currently is 'ipsec.1'
- **ip_address** (*str*) – Internet-routable IP address for customer's gateway. Must be a static address.
- **bgp_asn** (*str*) – Customer gateway's Border Gateway Protocol (BGP) Autonomous System Number (ASN)

Return type The newly created CustomerGateway

Returns A *boto.vpc.customergateway.CustomerGateway* object

create_dhcp_options(*vpc_id*, *cidr_block*, *availability_zone=None*)
Create a new DhcpOption

Parameters

- **vpc_id** (*str*) – The ID of the VPC where you want to create the subnet.
- **cidr_block** (*str*) – The CIDR block you want the subnet to cover.
- **availability_zone** (*str*) – The AZ you want the subnet in

Return type The newly created DhcpOption

Returns A `boto.vpc.customergateway.DhcpOption` object

create_internet_gateway()
Creates an internet gateway for VPC.

Return type Newly created internet gateway.

Returns *boto.vpc.internetgateway.InternetGateway*

create_route(*route_table_id*, *destination_cidr_block*, *gateway_id=None*, *instance_id=None*)
Creates a new route in the route table within a VPC. The route's target can be either a gateway attached to the VPC or a NAT instance in the VPC.

Parameters

- **route_table_id** (*str*) – The ID of the route table for the route.
- **destination_cidr_block** (*str*) – The CIDR address block used for the destination match.
- **gateway_id** (*str*) – The ID of the gateway attached to your VPC.
- **instance_id** (*str*) – The ID of a NAT instance in your VPC.

Return type bool

Returns True if successful

create_route_table (*vpc_id*)
Creates a new route table.

Parameters **vpc_id** (*str*) – The VPC ID to associate this route table with.

Return type The newly created route table

Returns A `boto.vpc.routetable.RouteTable` object

create_subnet (*vpc_id*, *cidr_block*, *availability_zone=None*)
Create a new Subnet

Parameters

- **vpc_id** (*str*) – The ID of the VPC where you want to create the subnet.
- **cidr_block** (*str*) – The CIDR block you want the subnet to cover.
- **availability_zone** (*str*) – The AZ you want the subnet in

Return type The newly created Subnet

Returns A `boto.vpc.customergateway.Subnet` object

create_vpc (*cidr_block*)
Create a new Virtual Private Cloud.

Parameters **cidr_block** (*str*) – A valid CIDR block

Return type The newly created VPC

Returns A `boto.vpc.vpc.VPC` object

create_vpn_connection (*type*, *customer_gateway_id*, *vpn_gateway_id*)
Create a new VPN Connection.

Parameters

- **type** (*str*) – The type of VPN Connection. Currently only 'ipsec.1' is supported
- **customer_gateway_id** (*str*) – The ID of the customer gateway.
- **vpn_gateway_id** (*str*) – The ID of the VPN gateway.

Return type The newly created VpnConnection

Returns A `boto.vpc.vpnconnection.VpnConnection` object

create_vpn_gateway (*type*, *availability_zone=None*)
Create a new Vpn Gateway

Parameters

- **type** (*str*) – Type of VPN Connection. Only valid valid currently is 'ipsec.1'

- **availability_zone** (*str*) – The Availability Zone where you want the VPN gateway.

Return type The newly created VpnGateway

Returns A *boto.vpc.vpngateway.VpnGateway* object

delete_customer_gateway(*customer_gateway_id*)
Delete a Customer Gateway.

Parameters **customer_gateway_id** (*str*) – The ID of the customer_gateway to be deleted.

Return type bool

Returns True if successful

delete_dhcp_options(*dhcp_options_id*)
Delete a DHCP Options

Parameters **dhcp_options_id** (*str*) – The ID of the DHCP Options to be deleted.

Return type bool

Returns True if successful

delete_internet_gateway(*internet_gateway_id*)
Deletes an internet gateway from the VPC.

Parameters **internet_gateway_id** (*str*) – The ID of the internet gateway to delete.

Return type Bool

Returns True if successful

delete_route(*route_table_id*, *destination_cidr_block*)
Deletes a route from a route table within a VPC.

Parameters

- **route_table_id** (*str*) – The ID of the route table with the route.
- **destination_cidr_block** (*str*) – The CIDR address block used for destination match.

Return type bool

Returns True if successful

delete_route_table(*route_table_id*)
Delete a route table.

Parameters **route_table_id** (*str*) – The ID of the route table to delete.

Return type bool

Returns True if successful

delete_subnet(*subnet_id*)
Delete a subnet.

Parameters **subnet_id** (*str*) – The ID of the subnet to be deleted.

Return type bool

Returns True if successful

delete_vpc(*vpc_id*)
Delete a Virtual Private Cloud.

Parameters **vpc_id** (*str*) – The ID of the vpc to be deleted.

Return type bool

Returns True if successful

delete_vpn_connection(*vpn_connection_id*)
Delete a VPN Connection.

Parameters **vpn_connection_id** (*str*) – The ID of the vpn_connection to be deleted.

Return type bool

Returns True if successful

delete_vpn_gateway(*vpn_gateway_id*)
Delete a Vpn Gateway.

Parameters **vpn_gateway_id** (*str*) – The ID of the vpn_gateway to be deleted.

Return type bool

Returns True if successful

detach_internet_gateway(*internet_gateway_id*, *vpc_id*)
Detach an internet gateway from a specific VPC.

Parameters

- **internet_gateway_id** (*str*) – The ID of the internet gateway to delete.
- **vpc_id** (*str*) – The ID of the VPC to attach to.

Return type Bool

Returns True if successful

disassociate_route_table(*association_id*)
Removes an association from a route table. This will cause all subnets that would've used this association to now use the main routing association instead.

Parameters **association_id** (*str*) – The ID of the association to disassociate.

Return type bool

Returns True if successful

get_all_customer_gateways(*customer_gateway_ids=None*, *filters=None*)
Retrieve information about your CustomerGateways. You can filter results to return information only about those CustomerGateways that match your search parameters. Otherwise, all CustomerGateways associated with your account are returned.

Parameters

- **customer_gateway_ids** (*list*) – A list of strings with the desired CustomerGateway ID's
- **filters** (*list of tuples*) – A list of tuples containing filters. Each tuple consists of a filter key and a filter value. Possible filter keys are:
  - *state*, the state of the CustomerGateway (pending,available,deleting,deleted)
  - *type*, the type of customer gateway (ipsec.1)
  - *ipAddress* the IP address of customer gateway's internet-routable external inteface

Return type *list*

Returns A list of `boto.vpc.customergateway.CustomerGateway`

get_all_dhcp_options(*dhcp_options_ids=None*)
Retrieve information about your DhcpOptions.

Parameters **dhcp_options_ids** (`list`) – A list of strings with the desired DhcpOption ID's

Return type *list*

Returns A list of `boto.vpc.dhcpoptions.DhcpOptions`

get_all_internet_gateways(*internet_gateway_ids=None*, *filters=None*)
Get a list of internet gateways. You can filter results to return information about only those gateways that you're interested in.

Parameters

- **internet_gateway_ids** (`list`) – A list of strings with the desired gateway IDs.
- **filters** (*list of tuples*) – A list of tuples containing filters. Each tuple consists of a filter key and a filter value.

get_all_route_tables(*route_table_ids=None*, *filters=None*)
Retrieve information about your routing tables. You can filter results to return information only about those route tables that match your search parameters. Otherwise, all route tables associated with your account are returned.

Parameters

- **route_table_ids** (`list`) – A list of strings with the desired route table IDs.
- **filters** (*list of tuples*) – A list of tuples containing filters. Each tuple consists of a filter key and a filter value.

Return type *list*

Returns A list of `boto.vpc.routetable.RouteTable`

get_all_subnets(*subnet_ids=None*, *filters=None*)
Retrieve information about your Subnets. You can filter results to return information only about those Subnets that match your search parameters. Otherwise, all Subnets associated with your account are returned.

Parameters

- **subnet_ids** (`list`) – A list of strings with the desired Subnet ID's
- **filters** (*list of tuples*) – A list of tuples containing filters. Each tuple consists of a filter key and a filter value. Possible filter keys are:
  - *state*, the state of the Subnet (pending,available)
  - *vpdId*, the ID of teh VPC the subnet is in.
  - *cidrBlock*, CIDR block of the subnet
  - *availabilityZone*, the Availability Zone the subnet is in.

Return type *list*

Returns A list of `boto.vpc.subnet.Subnet`

get_all_vpcs(*vpc_ids=None*, *filters=None*)
Retrieve information about your VPCs. You can filter results to return information only about those VPCs that match your search parameters. Otherwise, all VPCs associated with your account are returned.

Parameters

- **vpc_ids** (`list`) – A list of strings with the desired VPC ID's
- **filters** (*list of tuples*) – A list of tuples containing filters. Each tuple consists of a filter key and a filter value. Possible filter keys are:
  - *state*, the state of the VPC (pending or available)
  - *cidrBlock*, CIDR block of the VPC
  - *dhcpOptionsId*, the ID of a set of DHCP options

Return type *list*

Returns A list of `boto.vpc.vpc.VPC`

get_all_vpn_connections(*vpn_connection_ids=None*, *filters=None*)
Retrieve information about your VPN_CONNECTIONs. You can filter results to return information only about those VPN_CONNECTIONs that match your search parameters. Otherwise, all VPN_CONNECTIONs associated with your account are returned.

Parameters

- **vpn_connection_ids** (`list`) – A list of strings with the desired VPN_CONNECTION ID's
- **filters** (*list of tuples*) – A list of tuples containing filters. Each tuple consists of a filter key and a filter value. Possible filter keys are:
  - *state*, the state of the VPN_CONNECTION pending,available,deleting,deleted
  - *type*, the type of connection, currently 'ipsec.1'
  - *customerGatewayId*, the ID of the customer gateway associated with the VPN
  - *vpnGatewayId*, the ID of the VPN gateway associated with the VPN connection

Return type *list*

Returns A list of `boto.vpn_connection.vpnconnection.VpnConnection`

get_all_vpn_gateways(*vpn_gateway_ids=None*, *filters=None*)
Retrieve information about your VpnGateways. You can filter results to return information only about those VpnGateways that match your search parameters. Otherwise, all VpnGateways associated with your account are returned.

Parameters

- **vpn_gateway_ids** (`list`) – A list of strings with the desired VpnGateway ID's
- **filters** (*list of tuples*) – A list of tuples containing filters. Each tuple consists of a filter key and a filter value. Possible filter keys are:
  - *state*, the state of the VpnGateway (pending,available,deleting,deleted)
  - *type*, the type of customer gateway (ipsec.1)
  - *availabilityZone*, the Availability zone the VPN gateway is in.

Return type *list*

Returns A list of `boto.vpc.customergateway.VpnGateway`

boto.vpc.customergateway

Represents a Customer Gateway

class `boto.vpc.customergateway.`**`CustomerGateway`**(*connection=None*)

> **`endElement`**(*name, value, connection*)

boto.vpc.dhcpoptions

Represents a DHCP Options set

class `boto.vpc.dhcpoptions.`**`DhcpConfigSet`**

> **`endElement`**(*name, value, connection*)
>
> **`startElement`**(*name, attrs, connection*)

class `boto.vpc.dhcpoptions.`**`DhcpOptions`**(*connection=None*)

> **`endElement`**(*name, value, connection*)
>
> **`startElement`**(*name, attrs, connection*)

class `boto.vpc.dhcpoptions.`**`DhcpValueSet`**

> **`endElement`**(*name, value, connection*)
>
> **`startElement`**(*name, attrs, connection*)

boto.vpc.subnet

Represents a Subnet

class `boto.vpc.subnet.`**`Subnet`**(*connection=None*)

> **`endElement`**(*name, value, connection*)

boto.vpc.vpc

Represents a Virtual Private Cloud.

class `boto.vpc.vpc.`**`VPC`**(*connection=None*)

> **`delete`**()
>
> **`endElement`**(*name, value, connection*)

boto.vpc.vpnconnection

Represents a VPN Connectionn

class `boto.vpc.vpnconnection.`**`VpnConnection`**(*connection=None*)

> **`delete`**()
>
> **`endElement`**(*name*, *value*, *connection*)

boto.vpc.vpngateway

Represents a Vpn Gateway

class `boto.vpc.vpngateway.`**`Attachment`**(*connection=None*)

> **`endElement`**(*name*, *value*, *connection*)
>
> **`startElement`**(*name*, *attrs*, *connection*)

class `boto.vpc.vpngateway.`**`VpnGateway`**(*connection=None*)

> **`attach`**(*vpc_id*)
>
> **`endElement`**(*name*, *value*, *connection*)
>
> **`startElement`**(*name*, *attrs*, *connection*)

An Introduction to boto's Elastic Load Balancing interface

This tutorial focuses on the boto interface for Elastic Load Balancing from Amazon Web Services. This tutorial assumes that you have already downloaded and installed boto, and are familiar with the boto ec2 interface.

Elastic Load Balancing Concepts

Elastic Load Balancing (ELB) is intimately connected with Amazon's Elastic Compute Cloud (EC2) service. Using the ELB service allows you to create a load balancer - a DNS endpoint and set of ports that distributes incoming requests to a set of EC2 instances. The advantages of using a load balancer is that it allows you to truly scale up or down a set of backend instances without disrupting service. Before the ELB service, you had to do this manually by launching an EC2 instance and installing load balancer software on it (nginx, haproxy, perlbal, etc.) to distribute traffic to other EC2 instances.

Recall that the EC2 service is split into Regions, which are further divided into Availability Zones (AZ). For example, the US-East region is divided into us-east-1a, us-east-1b, us-east-1c, us-east-1d, and us-east-1e. You can think of AZs as data centers - each runs off a different set of ISP backbones and power providers. ELB load balancers can span multiple AZs but cannot span multiple regions. That means that if you'd like to create a set of instances spanning both the US and Europe Regions you'd have to create two load balancers and have some sort of other means of distributing requests between the two load balancers. An example of this could be using GeoIP techniques to choose the correct load balancer, or perhaps DNS round robin. Keep in mind also that traffic is distributed equally over all AZs the ELB balancer spans. This means you should have an equal number of instances in each AZ if you want to equally distribute load amongst all your instances.

Creating a Connection

The first step in accessing ELB is to create a connection to the service.

```
>>> import boto
>>> conn = boto.connect_elb(
        aws_access_key_id='YOUR-KEY-ID-HERE',
        aws_secret_access_key='YOUR-SECRET-HERE'
    )
```

A Note About Regions and Endpoints

Like EC2, the ELB service has a different endpoint for each region. By default the US East endpoint is used. To choose a specific region, instantiate the ELBConnection object with that region's information.

```
>>> from boto.regioninfo import RegionInfo
>>> reg = RegionInfo(
        name='eu-west-1',
        endpoint='elasticloadbalancing.eu-west-1.amazonaws.com'
    )
>>> conn = boto.connect_elb(
        aws_access_key_id='YOUR-KEY-ID-HERE',
        aws_secret_access_key='YOUR-SECRET-HERE',
        region=reg
    )
```

Another way to connect to an alternative region is like this:

```
>>> import boto.ec2.elb
>>> elb = boto.ec2.elb.connect_to_region('eu-west-1')
```

Here's yet another way to discover what regions are available and then connect to one:

```
>>> import boto.ec2.elb
>>> regions = boto.ec2.elb.regions()
>>> regions
[RegionInfo:us-east-1,
 RegionInfo:ap-northeast-1,
 RegionInfo:us-west-1,
 RegionInfo:ap-southeast-1,
 RegionInfo:eu-west-1]
>>> elb = regions[-1].connect()
```

Alternatively, edit your boto.cfg with the default ELB endpoint to use:

```
[Boto]
elb_region_name = eu-west-1
elb_region_endpoint = elasticloadbalancing.eu-west-1.amazonaws.com
```

Getting Existing Load Balancers

To retrieve any exiting load balancers:

```
>>> conn.get_all_load_balancers()
[LoadBalancer:load-balancer-prod, LoadBalancer:load-balancer-staging]
```

You can also filter by name

```
>>> conn.get_all_load_balancers(load_balancer_names=['load-balancer-prod'])
[LoadBalancer:load-balancer-prod]
```

`get_all_load_balancers` returns a `boto.resultset.ResultSet` that contains instances of `boto.ec2.elb.loadbalancer.LoadBalancer`, each of which abstracts access to a load balancer. `ResultSet` works very much like a list.

```
>>> balancers = conn.get_all_load_balancers()
>>> balancers[0]
[LoadBalancer:load-balancer-prod]
```

Creating a Load Balancer

To create a load balancer you need the following:

1. The specific **ports and protocols** you want to load balancer over, and what port you want to connect to all instances.
2. A **health check** - the ELB concept of a *heart beat* or *ping*. ELB will use this health check to see whether your instances are up or down. If they go down, the load balancer will no longer send requests to them.
3. A **list of Availability Zones** you'd like to create your load balancer over.

Ports and Protocols

An incoming connection to your load balancer will come on one or more ports - for example 80 (HTTP) and 443 (HTTPS). Each can be using a protocol - currently, the supported protocols are TCP and HTTP. We also need to tell the load balancer which port to route connects *to* on each instance. For example, to create a load balancer for a website that accepts connections on 80 and 443, and that routes connections to port 8080 and 8443 on each instance, you would specify that the load balancer ports and protocols are:

- 80, 8080, HTTP
- 443, 8443, TCP

This says that the load balancer will listen on two ports - 80 and 443. Connections on 80 will use an HTTP load balancer to forward connections to port 8080 on instances. Likewise, the load balancer will listen on 443 to forward connections to 8443 on each instance using the TCP balancer. We need to use TCP for the HTTPS port because it is encrypted at the application layer. Of course, we could specify the load balancer use TCP for port 80, however specifying HTTP allows you to let ELB handle some work for you - for example HTTP header parsing.

Configuring a Health Check

A health check allows ELB to determine which instances are alive and able to respond to requests. A health check is essentially a tuple consisting of:

- *Target*: What to check on an instance. For a TCP check this is comprised of:

  ```
  TCP:PORT_TO_CHECK
  ```

  Which attempts to open a connection on PORT_TO_CHECK. If the connection opens successfully, that specific instance is deemed healthy, otherwise it is marked temporarily as unhealthy. For HTTP, the situation is slightly different:

```
HTTP:PORT_TO_CHECK/RESOURCE
```

This means that the health check will connect to the resource /RESOURCE on PORT_TO_CHECK. If an HTTP 200 status is returned the instance is deemed healthy.

- *Interval*: How often the check is made. This is given in seconds and defaults to 30. The valid range of intervals goes from 5 seconds to 600 seconds.
- *Timeout*: The number of seconds the load balancer will wait for a check to return a result.
- *Unhealthy threshold*: The number of consecutive failed checks to deem the instance as being dead. The default is 5, and the range of valid values lies from 2 to 10.

The following example creates a health check called *instance_health* that simply checks instances every 20 seconds on port 80 over HTTP at the resource /health for 200 successes.

```
>>> from boto.ec2.elb import HealthCheck
>>> hc = HealthCheck(
        interval=20,
        healthy_threshold=3,
        unhealthy_threshold=5,
        target='HTTP:8080/health'
    )
```

Putting It All Together

Finally, let's create a load balancer in the US region that listens on ports 80 and 443 and distributes requests to instances on 8080 and 8443 over HTTP and TCP. We want the load balancer to span the availability zones *us-east-1a* and *us-east-1b*:

```
>>> regions = ['us-east-1a', 'us-east-1b']
>>> ports = [(80, 8080, 'http'), (443, 8443, 'tcp')]
>>> lb = conn.create_load_balancer('my-lb', regions, ports)
>>> # This is from the previous section.
>>> lb.configure_health_check(hc)
```

The load balancer has been created. To see where you can actually connect to it, do:

```
>>> print lb.dns_name
my_elb-123456789.us-east-1.elb.amazonaws.com
```

You can then CNAME map a better name, i.e. www.MYWEBSITE.com to the above address.

Adding Instances To a Load Balancer

Now that the load balancer has been created, there are two ways to add instances to it:

1. Manually, adding each instance in turn.
2. Mapping an autoscale group to the load balancer. Please see the *Autoscale tutorial* for information on how to do this.

Manually Adding and Removing Instances

Assuming you have a list of instance ids, you can add them to the load balancer

```
>>> instance_ids = ['i-4f8cf126', 'i-0bb7ca62']
>>> lb.register_instances(instance_ids)
```

Keep in mind that these instances should be in Security Groups that match the internal ports of the load balancer you just created (for this example, they should allow incoming connections on 8080 and 8443).

To remove instances:

```
>>> lb.degregister_instances(instance_ids)
```

Modifying Availability Zones for a Load Balancer

If you wanted to disable one or more zones from an existing load balancer:

```
>>> lb.disable_zones(['us-east-1a'])
```

You can then terminate each instance in the disabled zone and then deregister then from your load balancer.

To enable zones:

```
>>> lb.enable_zones(['us-east-1c'])
```

Deleting a Load Balancer

```
>>> lb.delete()
```

ELB Reference

boto.ec2.elb

This module provides an interface to the Elastic Compute Cloud (EC2) load balancing service from AWS.

class `boto.ec2.elb.`**`ELBConnection`**(*aws_access_key_id=None*, *aws_secret_access_key=None*, *is_secure=False*, *port=None*, *proxy=None*, *proxy_port=None*, *proxy_user=None*, *proxy_pass=None*, *debug=0*, *https_connection_factory=None*, *region=None*, *path='/'*)

Init method to create a new connection to EC2 Load Balancing Service.

Note: The region argument is overridden by the region specified in the boto configuration file.

`APIVersion` = **'2011-11-15'**

`DefaultRegionEndpoint` = **'elasticloadbalancing.amazonaws.com'**

`DefaultRegionName` = **'us-east-1'**

`apply_security_groups_to_lb`(*name*, *security_groups*)

Applies security groups to the load balancer. Applying security groups that are already registered with the Load Balancer has no effect.

Parameters

- **name** (*string*) – The name of the Load Balancer
- **security_groups** (*List of strings*) – The name of the security group(s) to add.

Return type List of strings

Returns An updated list of security groups for this Load Balancer.

attach_lb_to_subnets(*name*, *subnets*)
Attaches load balancer to one or more subnets. Attaching subnets that are already registered with the Load Balancer has no effect.

Parameters

- **name** (*string*) – The name of the Load Balancer
- **subnets** (*List of strings*) – The name of the subnet(s) to add.

Return type List of strings

Returns An updated list of subnets for this Load Balancer.

build_list_params(*params*, *items*, *label*)

configure_health_check(*name*, *health_check*)
Define a health check for the EndPoints.

Parameters

- **name** (*string*) – The mnemonic name associated with the load balancer
- **health_check** (*boto.ec2.elb.healthcheck.HealthCheck*) – A HealthCheck object populated with the desired values.

Return type *boto.ec2.elb.healthcheck.HealthCheck*

Returns The updated *boto.ec2.elb.healthcheck.HealthCheck*

create_app_cookie_stickiness_policy(*name*, *lb_name*, *policy_name*)
Generates a stickiness policy with sticky session lifetimes that follow that of an application-generated cookie. This policy can only be associated with HTTP listeners.

This policy is similar to the policy created by CreateLBCookieStickinessPolicy, except that the lifetime of the special Elastic Load Balancing cookie follows the lifetime of the application-generated cookie specified in the policy configuration. The load balancer only inserts a new stickiness cookie when the application response includes a new application cookie.

If the application cookie is explicitly removed or expires, the session stops being sticky until a new application cookie is issued.

create_lb_cookie_stickiness_policy(*cookie_expiration_period*, *lb_name*, *policy_name*)
Generates a stickiness policy with sticky session lifetimes controlled by the lifetime of the browser (user-agent) or a specified expiration period. This policy can only be associated only with HTTP listeners.

When a load balancer implements this policy, the load balancer uses a special cookie to track the backend server instance for each request. When the load balancer receives a request, it first checks to see if this cookie is present in the request. If so, the load balancer sends the request to the application server specified in the cookie. If not, the load balancer sends the request to a server that is chosen based on the existing load balancing algorithm.

A cookie is inserted into the response for binding subsequent requests from the same user to that server. The validity of the cookie is based on the cookie expiration time, which is specified in the policy configuration.

create_load_balancer(*name*, *zones*, *listeners*, *subnets=None*, *security_groups=None*)
Create a new load balancer for your account. By default the load balancer will be created in EC2. To create a load balancer inside a VPC, parameter zones must be set to None and subnets must not be None. The load balancer will be automatically created under the VPC that contains the subnet(s) specified.

Parameters

- **name** (*string*) – The mnemonic name associated with the new load balancer
- **zones** (*List of strings*) – The names of the availability zone(s) to add.
- **listeners** (*List of tuples*) – Each tuple contains three or four values, (LoadBalancerPortNumber, InstancePortNumber, Protocol, [SSLCertificateId]) where LoadBalancerPortNumber and InstancePortNumber are integer values between 1 and 65535, Protocol is a string containing either 'TCP', 'HTTP' or 'HTTPS'; SSLCertificateID is the ARN of a AWS AIM certificate, and must be specified when doing HTTPS.

Return type *boto.ec2.elb.loadbalancer.LoadBalancer*

Returns The newly created *boto.ec2.elb.loadbalancer.LoadBalancer*

create_load_balancer_listeners(*name*, *listeners*)
Creates a Listener (or group of listeners) for an existing Load Balancer

Parameters

- **name** (*string*) – The name of the load balancer to create the listeners for
- **listeners** (*List of tuples*) – Each tuple contains three values, (LoadBalancerPortNumber, InstancePortNumber, Protocol, [SSLCertificateId]) where LoadBalancerPortNumber and InstancePortNumber are integer values between 1 and 65535, Protocol is a string containing either 'TCP', 'HTTP' or 'HTTPS'; SSLCertificateID is the ARN of a AWS AIM certificate, and must be specified when doing HTTPS.

Returns The status of the request

delete_lb_policy(*lb_name*, *policy_name*)
Deletes a policy from the LoadBalancer. The specified policy must not be enabled for any listeners.

delete_load_balancer(*name*)
Delete a Load Balancer from your account.

Parameters **name** (*string*) – The name of the Load Balancer to delete

delete_load_balancer_listeners(*name*, *ports*)
Deletes a load balancer listener (or group of listeners)

Parameters

- **name** (*string*) – The name of the load balancer to create the listeners for
- **ports** (*List int*) – Each int represents the port on the ELB to be removed

Returns The status of the request

deregister_instances(*load_balancer_name*, *instances*)
Remove Instances from an existing Load Balancer.

Parameters

- **load_balancer_name** (*string*) – The name of the Load Balancer
- **instances** (*List of strings*) – The instance ID's of the EC2 instances to remove.

Return type List of strings

Returns An updated list of instances for this Load Balancer.

describe_instance_health(*load_balancer_name*, *instances=None*)
Get current state of all Instances registered to an Load Balancer.

Parameters

- **load_balancer_name** (*string*) – The name of the Load Balancer
- **instances** (*List of strings*) – The instance ID's of the EC2 instances to return status for. If not provided, the state of all instances will be returned.

Return type List of *boto.ec2.elb.instancestate.InstanceState*

Returns list of state info for instances in this Load Balancer.

detach_lb_from_subnets(*name*, *subnets*)
Detaches load balancer from one or more subnets.

Parameters

- **name** (*string*) – The name of the Load Balancer
- **subnets** (*List of strings*) – The name of the subnet(s) to detach.

Return type List of strings

Returns An updated list of subnets for this Load Balancer.

disable_availability_zones(*load_balancer_name*, *zones_to_remove*)
Remove availability zones from an existing Load Balancer. All zones must be in the same region as the Load Balancer. Removing zones that are not registered with the Load Balancer has no effect. You cannot remove all zones from an Load Balancer.

Parameters

- **load_balancer_name** (*string*) – The name of the Load Balancer
- **zones** (*List of strings*) – The name of the zone(s) to remove.

Return type List of strings

Returns An updated list of zones for this Load Balancer.

enable_availability_zones(*load_balancer_name*, *zones_to_add*)
Add availability zones to an existing Load Balancer All zones must be in the same region as the Load Balancer Adding zones that are already registered with the Load Balancer has no effect.

Parameters

- **load_balancer_name** (*string*) – The name of the Load Balancer
- **zones** (*List of strings*) – The name of the zone(s) to add.

Return type List of strings

Returns An updated list of zones for this Load Balancer.

get_all_load_balancers(*load_balancer_names=None*)
Retrieve all load balancers associated with your account.

Parameters **load_balancer_names** (*list*) – An optional list of load balancer names.

Return type *boto.resultset.ResultSet*

Returns A ResultSet containing instances of *boto.ec2.elb.loadbalancer.LoadBalancer*

register_instances(*load_balancer_name*, *instances*)
Add new Instances to an existing Load Balancer.

Parameters

- **load_balancer_name** (*string*) – The name of the Load Balancer
- **instances** (*List of strings*) – The instance ID's of the EC2 instances to add.

Return type List of strings

Returns An updated list of instances for this Load Balancer.

set_lb_listener_SSL_certificate(*lb_name*, *lb_port*, *ssl_certificate_id*)
Sets the certificate that terminates the specified listener's SSL connections. The specified certificate replaces any prior certificate that was used on the same LoadBalancer and port.

set_lb_policies_of_listener(*lb_name*, *lb_port*, *policies*)
Associates, updates, or disables a policy with a listener on the load balancer. Currently only zero (0) or one (1) policy can be associated with a listener.

`boto.ec2.elb.`**connect_to_region**(*region_name*, ***kw_params*)
Given a valid region name, return a *boto.ec2.elb.ELBConnection*.

Parameters **region_name** (*str*) – The name of the region to connect to.

Return type `boto.ec2.ELBConnection` or `None`

Returns A connection to the given region, or None if an invalid region name is given

`boto.ec2.elb.`**regions**()
Get all available regions for the SDB service.

Return type *list*

Returns A list of `boto.RegionInfo` instances

boto.ec2.elb.healthcheck

class `boto.ec2.elb.healthcheck.`**HealthCheck**(*access_point=None*, *interval=30*, *target=None*, *healthy_threshold=3*, *timeout=5*, *unhealthy_threshold=5*)
Represents an EC2 Access Point Health Check. See *Configuring a Health Check* for a walkthrough on configuring load balancer health checks.

Variables

- **access_point** (*str*) – The name of the load balancer this health check is associated with.
- **interval** (*int*) – Specifies how many seconds there are between health checks.
- **target** (*str*) – Determines what to check on an instance. See the Amazon HealthCheck documentation for possible Target values.

endElement(*name*, *value*, *connection*)

startElement(*name*, *attrs*, *connection*)

update()
In the case where you have accessed an existing health check on a load balancer, this method applies this instance's health check values to the load balancer it is attached to.

Note: This method will not do anything if the `access_point` attribute isn't set, as is the case with a newly instantiated HealthCheck instance.

boto.ec2.elb.instancestate

class `boto.ec2.elb.instancestate.`**`InstanceState`**(*load_balancer=None*, *description=None*, *state=None*, *instance_id=None*, *reason_code=None*)

Represents the state of an EC2 Load Balancer Instance

Variables

- **`load_balancer`** (`boto.ec2.elb.loadbalancer.LoadBalancer`) – The load balancer this instance is registered to.
- ***`description`*** (*`str`*) – A description of the instance.
- ***`instance_id`*** (*`str`*) – The EC2 instance ID.
- **`reason_code`** (*`str`*) – Provides information about the cause of an OutOfService instance. Specifically, it indicates whether the cause is Elastic Load Balancing or the instance behind the LoadBalancer.
- **`state`** (*`str`*) – Specifies the current state of the instance.

`endElement`(*name*, *value*, *connection*)

`startElement`(*name*, *attrs*, *connection*)

boto.ec2.elb.listelement

class `boto.ec2.elb.listelement.`**`ListElement`**

A `list` subclass that has some additional methods for interacting with Amazon's XML API.

`endElement`(*name*, *value*, *connection*)

`startElement`(*name*, *attrs*, *connection*)

boto.ec2.elb.listener

class `boto.ec2.elb.listener.`**`Listener`**(*load_balancer=None*, *load_balancer_port=0*, *instance_port=0*, *protocol=''*, *ssl_certificate_id=None*)

Represents an EC2 Load Balancer Listener tuple

`endElement`(*name*, *value*, *connection*)

`get_tuple`()

`startElement`(*name*, *attrs*, *connection*)

boto.ec2.elb.loadbalancer

class `boto.ec2.elb.loadbalancer.`**`LoadBalancer`**(*connection=None*, *name=None*, *endpoints=None*)

Represents an EC2 Load Balancer.

Variables

- ***connection*** (boto.ec2.elb.ELBConnection) – The connection this load balancer was instance was instantiated from.
- **listeners** (list) – A list of tuples in the form of (<Inbound port>, <Outbound port>, <Protocol>)
- **health_check** (boto.ec2.elb.healthcheck.HealthCheck) – The health check policy for this load balancer.
- **policies** (*boto.ec2.elb.policies.Policies*) – Cookie stickiness and other policies.
- **dns_name** (*str*) – The external DNS name for the balancer.
- **created_time** (*str*) – A date+time string showing when the load balancer was created.
- ***instances*** (list) – A list of *boto.ec2.instanceinfo.InstanceInfo* instances, representing the EC2 instances this load balancer is distributing requests to.
- **availability_zones** (list) – The availability zones this balancer covers.
- **canonical_hosted_zone_name** (*str*) – Current CNAME for the balancer.
- **canonical_hosted_zone_name_id** (*str*) – The Route 53 hosted zone ID of this balancer. Needed when creating an Alias record in a Route 53 hosted zone.
- **source_security_group** (*boto.ec2.elb.securitygroup.SecurityGroup*) – The security group that you can use as part of your inbound rules for your load balancer back-end instances to disallow traffic from sources other than your load balancer.
- **subnets** (list) – A list of subnets this balancer is on.
- **security_groups** (list) – A list of additional security groups that have been applied.
- **vpc_id** (*str*) – The ID of the VPC that this ELB resides within.

apply_security_groups(*security_groups*)

Applies security groups to the load balancer. Applying security groups that are already registered with the Load Balancer has no effect.

Parameters security_groups (*string or List of strings*) – The name of the security group(s) to add.

attach_subnets(*subnets*)

Attaches load balancer to one or more subnets. Attaching subnets that are already registered with the Load Balancer has no effect.

Parameters subnets (*string or List of strings*) – The name of the subnet(s) to add.

configure_health_check(*health_check*)

Configures the health check behavior for the instances behind this load balancer. See *Configuring a Health Check* for a walkthrough.

Parameters health_check (boto.ec2.elb.healthcheck.HealthCheck) – A HealthCheck instance that tells the load balancer how to check its instances for health.

create_app_cookie_stickiness_policy(*name*, *policy_name*)

create_cookie_stickiness_policy(*cookie_expiration_period*, *policy_name*)

create_listener(*inPort*, *outPort=None*, *proto='tcp'*)

create_listeners(*listeners*)

delete()
Delete this load balancer.

delete_listener(*inPort*)

delete_listeners(*listeners*)

delete_policy(*policy_name*)
Deletes a policy from the LoadBalancer. The specified policy must not be enabled for any listeners.

deregister_instances(*instances*)
Remove instances from this load balancer. Removing instances that are not registered with the load balancer has no effect.

Parameters **instances** (`list`) – List of instance IDs (strings) that you'd like to remove from this load balancer.

detach_subnets(*subnets*)
Detaches load balancer from one or more subnets.

Parameters **subnets** (*string or List of strings*) – The name of the subnet(s) to detach.

disable_zones(*zones*)
Disable availability zones from this Access Point.

Parameters **zones** (*string or List of strings*) – The name of the zone(s) to add.

enable_zones(*zones*)
Enable availability zones to this Access Point. All zones must be in the same region as the Access Point.

Parameters **zones** (*string or List of strings*) – The name of the zone(s) to add.

endElement(*name*, *value*, *connection*)

get_instance_health(*instances=None*)
Returns a list of *boto.ec2.elb.instancestate.InstanceState* objects, which show the health of the instances attached to this load balancer.

Return type *list*

Returns A list of *InstanceState* instances, representing the instances attached to this load balancer.

register_instances(*instances*)
Adds instances to this load balancer. All instances must be in the same region as the load balancer. Adding endpoints that are already registered with the load balancer has no effect.

Parameters **instances** (`list`) – List of instance IDs (strings) that you'd like to add to this load balancer.

set_listener_SSL_certificate(*lb_port*, *ssl_certificate_id*)

set_policies_of_listener(*lb_port*, *policies*)

startElement(*name*, *attrs*, *connection*)

fps

boto.fps

boto.fps.connection

class `boto.fps.connection.`**`FPSConnection`**(*aws_access_key_id=None, aws_secret_access_key=None, is_secure=True, port=None, proxy=None, proxy_port=None, proxy_user=None, proxy_pass=None, host='fps.sandbox.amazonaws.com', debug=0, https_connection_factory=None, path='/'*)

`APIVersion` = **'2007-01-08'**

`cancel`(*transactionId, description=None*)
Cancels a reserved or pending transaction.

`get_recipient_verification_status`(*recipientTokenId*)
Test that the intended recipient has a verified Amazon Payments account.

`get_token_by_caller_reference`(*callerReference*)
Returns details about the token specified by 'CallerReference'.

`get_token_by_caller_token`(*tokenId*)
Returns details about the token specified by 'TokenId'.

`get_transaction_status`(*transactionId*)
Returns the status of a given transaction.

`install_caller_instruction`(*token_type='Unrestricted', transaction_id=None*)
Set us up as a caller This will install a new caller_token into the FPS section. This should really only be called to regenerate the caller token.

`install_payment_instruction`(*instruction, token_type='Unrestricted', transaction_id=None*)
InstallPaymentInstruction instruction: The PaymentInstruction to send, for example:

MyRole=='Caller' orSay 'Roles do not match';

token_type: Defaults to "Unrestricted" transaction_id: Defaults to a new ID

`install_recipient_instruction`(*token_type='Unrestricted', transaction_id=None*)
Set us up as a Recipient This will install a new caller_token into the FPS section. This should really only be called to regenerate the recipient token.

`make_marketplace_registration_url`(*returnURL, pipelineName, maxFixedFee=0.0, maxVariableFee=0.0, recipientPaysFee=True, **params*)
Generate the URL with the signature required for signing up a recipient

`make_url`(*returnURL, paymentReason, pipelineName, transactionAmount, **params*)
Generate the URL with the signature required for a transaction

`pay`(*transactionAmount, senderTokenId, recipientTokenId=None, callerTokenId=None, chargeFeeTo='Recipient', callerReference=None, senderReference=None, recipientReference=None, senderDescription=None, recipientDescription=None, callerDescription=None, metadata=None, transactionDate=None, reserve=False*)
Make a payment transaction. You must specify the amount. This can also perform a Reserve request if 'reserve' is set to True.

refund(*callerReference*, *transactionId*, *refundAmount=None*, *callerDescription=None*)
Refund a transaction. This refunds the full amount by default unless 'refundAmount' is specified.

settle(*reserveTransactionId*, *transactionAmount=None*)
Charges for a reserved payment.

verify_signature(*end_point_url*, *http_parameters*)

An Introduction to boto's S3 interface

This tutorial focuses on the boto interface to the Simple Storage Service from Amazon Web Services. This tutorial assumes that you have already downloaded and installed boto.

Creating a Connection

The first step in accessing S3 is to create a connection to the service. There are two ways to do this in boto. The first is:

```
>>> from boto.s3.connection import S3Connection
>>> conn = S3Connection('<aws access key>', '<aws secret key>')
```

At this point the variable conn will point to an S3Connection object. In this example, the AWS access key and AWS secret key are passed in to the method explicitely. Alternatively, you can set the environment variables:

AWS_ACCESS_KEY_ID - Your AWS Access Key ID AWS_SECRET_ACCESS_KEY - Your AWS Secret Access Key

and then call the constructor without any arguments, like this:

```
>>> conn = S3Connection()
```

There is also a shortcut function in the boto package, called connect_s3 that may provide a slightly easier means of creating a connection:

```
>>> import boto
>>> conn = boto.connect_s3()
```

In either case, conn will point to an S3Connection object which we will use throughout the remainder of this tutorial.

Creating a Bucket

Once you have a connection established with S3, you will probably want to create a bucket. A bucket is a container used to store key/value pairs in S3. A bucket can hold an unlimited amount of data so you could potentially have just one bucket in S3 for all of your information. Or, you could create separate buckets for different types of data. You can figure all of that out later, first let's just create a bucket. That can be accomplished like this:

```
>>> bucket = conn.create_bucket('mybucket')
Traceback (most recent call last):
  File "<stdin>", line 1, in ?
  File "boto/connection.py", line 285, in create_bucket
    raise S3CreateError(response.status, response.reason)
boto.exception.S3CreateError: S3Error[409]: Conflict
```

Whoa. What happended there? Well, the thing you have to know about buckets is that they are kind of like domain names. It's one flat name space that everyone who uses S3 shares. So, someone has already create a bucket called "mybucket" in S3 and that means no one else can grab that bucket name. So, you have to come up with a name that hasn't been taken yet. For example, something that uses a unique string as a prefix. Your AWS_ACCESS_KEY (NOT YOUR SECRET KEY!) could work but I'll leave it to your imagination to come up with something. I'll just assume that you found an acceptable name.

The create_bucket method will create the requested bucket if it does not exist or will return the existing bucket if it does exist.

Creating a Bucket In Another Location

The example above assumes that you want to create a bucket in the standard US region. However, it is possible to create buckets in other locations. To do so, first import the Location object from the boto.s3.connection module, like this:

```
>>> from boto.s3.connection import Location
>>> dir(Location)
['DEFAULT', 'EU', 'USWest', 'APSoutheast', '__doc__', '__module__']
>>>
```

As you can see, the Location object defines three possible locations; DEFAULT, EU, USWest, and APSoutheast. By default, the location is the empty string which is interpreted as the US Classic Region, the original S3 region. However, by specifying another location at the time the bucket is created, you can instruct S3 to create the bucket in that location. For example:

```
>>> conn.create_bucket('mybucket', location=Location.EU)
```

will create the bucket in the EU region (assuming the name is available).

Storing Data

Once you have a bucket, presumably you will want to store some data in it. S3 doesn't care what kind of information you store in your objects or what format you use to store it. All you need is a key that is unique within your bucket.

The Key object is used in boto to keep track of data stored in S3. To store new data in S3, start by creating a new Key object:

```
>>> from boto.s3.key import Key
>>> k = Key(bucket)
>>> k.key = 'foobar'
>>> k.set_contents_from_string('This is a test of S3')
```

The net effect of these statements is to create a new object in S3 with a key of "foobar" and a value of "This is a test of S3". To validate that this worked, quit out of the interpreter and start it up again. Then:

```
>>> import boto
>>> c = boto.connect_s3()
>>> b = c.create_bucket('mybucket') # substitute your bucket name here
>>> from boto.s3.key import Key
>>> k = Key(b)
>>> k.key = 'foobar'
>>> k.get_contents_as_string()
'This is a test of S3'
```

So, we can definitely store and retrieve strings. A more interesting example may be to store the contents of a local file in S3 and then retrieve the contents to another local file.

```
>>> k = Key(b)
>>> k.key = 'myfile'
>>> k.set_contents_from_filename('foo.jpg')
>>> k.get_contents_to_filename('bar.jpg')
```

There are a couple of things to note about this. When you send data to S3 from a file or filename, boto will attempt to determine the correct mime type for that file and send it as a Content-Type header. The boto package uses the standard mimetypes package in Python to do the mime type guessing. The other thing to note is that boto does stream the content to and from S3 so you should be able to send and receive large files without any problem.

Listing All Available Buckets

In addition to accessing specific buckets via the create_bucket method you can also get a list of all available buckets that you have created.

```
>>> rs = conn.get_all_buckets()
```

This returns a ResultSet object (see the SQS Tutorial for more info on ResultSet objects). The ResultSet can be used as a sequence or list type object to retrieve Bucket objects.

```
>>> len(rs)
11
>>> for b in rs:
... print b.name
...
<listing of available buckets>
>>> b = rs[0]
```

Setting / Getting the Access Control List for Buckets and Keys

The S3 service provides the ability to control access to buckets and keys within s3 via the Access Control List (ACL) associated with each object in S3. There are two ways to set the ACL for an object:

1. Create a custom ACL that grants specific rights to specific users. At the moment, the users that are specified within grants have to be registered users of Amazon Web Services so this isn't as useful or as general as it could be.
2. Use a "canned" access control policy. There are four canned policies defined: a. private: Owner gets FULL_CONTROL. No one else has any access rights. b. public-read: Owners gets FULL_CONTROL and the anonymous principal is granted READ access. c. public-read-write: Owner gets FULL_CONTROL and the anonymous principal is granted READ and WRITE access. d. authenticated-read: Owner gets FULL_CONTROL and any principal authenticated as a registered Amazon S3 user is granted READ access.

To set a canned ACL for a bucket, use the set_acl method of the Bucket object. The argument passed to this method must be one of the four permissable canned policies named in the list CannedACLStrings contained in acl.py. For example, to make a bucket readable by anyone:

```
>>> b.set_acl('public-read')
```

You can also set the ACL for Key objects, either by passing an additional argument to the above method:

```
>>> b.set_acl('public-read', 'foobar')
```

where ‘foobar’ is the key of some object within the bucket b or you can call the set_acl method of the Key object:

```
>>> k.set_acl('public-read')
```

You can also retrieve the current ACL for a Bucket or Key object using the get_acl object. This method parses the AccessControlPolicy response sent by S3 and creates a set of Python objects that represent the ACL.

```
>>> acp = b.get_acl()
>>> acp
<boto.acl.Policy instance at 0x2e6940>
>>> acp.acl
<boto.acl.ACL instance at 0x2e69e0>
>>> acp.acl.grants
[<boto.acl.Grant instance at 0x2e6a08>]
>>> for grant in acp.acl.grants:
...   print grant.permission, grant.display_name, grant.email_address, grant.id
...
FULL_CONTROL <boto.user.User instance at 0x2e6a30>
```

The Python objects representing the ACL can be found in the acl.py module of boto.

Both the Bucket object and the Key object also provide shortcut methods to simplify the process of granting individuals specific access. For example, if you want to grant an individual user READ access to a particular object in S3 you could do the following:

```
>>> key = b.lookup('mykeytoshare')
>>> key.add_email_grant('READ', 'foo@bar.com')
```

The email address provided should be the one associated with the users AWS account. There is a similar method called add_user_grant that accepts the canonical id of the user rather than the email address.

Setting/Getting Metadata Values on Key Objects

S3 allows arbitrary user metadata to be assigned to objects within a bucket. To take advantage of this S3 feature, you should use the set_metadata and get_metadata methods of the Key object to set and retrieve metadata associated with an S3 object. For example:

```
>>> k = Key(b)
>>> k.key = 'has_metadata'
>>> k.set_metadata('meta1', 'This is the first metadata value')
>>> k.set_metadata('meta2', 'This is the second metadata value')
>>> k.set_contents_from_filename('foo.txt')
```

This code associates two metadata key/value pairs with the Key k. To retrieve those values later:

```
>>> k = b.get_key('has_metadata)
>>> k.get_metadata('meta1')
'This is the first metadata value'
>>> k.get_metadata('meta2')
'This is the second metadata value'
>>>
```

S3

boto.s3.acl

class `boto.s3.acl.`**`ACL`**(*policy=None*)

> **`add_email_grant`**(*permission*, *email_address*)
>
> **`add_grant`**(*grant*)
>
> **`add_user_grant`**(*permission*, *user_id*, *display_name=None*)
>
> **`endElement`**(*name*, *value*, *connection*)
>
> **`startElement`**(*name*, *attrs*, *connection*)
>
> **`to_xml`**()

class `boto.s3.acl.`**`Grant`**(*permission=None*, *type=None*, *id=None*, *display_name=None*, *uri=None*, *email_address=None*)

> **`NameSpace`** = **'xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"'**
>
> **`endElement`**(*name*, *value*, *connection*)
>
> **`startElement`**(*name*, *attrs*, *connection*)
>
> **`to_xml`**()

class `boto.s3.acl.`**`Policy`**(*parent=None*)

> **`endElement`**(*name*, *value*, *connection*)
>
> **`startElement`**(*name*, *attrs*, *connection*)
>
> **`to_xml`**()

boto.s3.bucket

class `boto.s3.bucket.`**`Bucket`**(*connection=None*, *name=None*, *key_class=<class 'boto.s3.key.Key'>*)

> **`BucketLoggingBody`** = **'<?xml version="1.0" encoding="UTF-8"?>\n <BucketLoggingStatus xmlns="http://s3.amazon**
>
> **`BucketPaymentBody`** = **'<?xml version="1.0" encoding="UTF-8"?>\n <RequestPaymentConfiguration xmlns="http://s**
>
> **`EmptyBucketLoggingBody`** = **'<?xml version="1.0" encoding="UTF-8"?>\n <BucketLoggingStatus xmlns="http://s3.a**
>
> **`LoggingGroup`** = **'http://acs.amazonaws.com/groups/s3/LogDelivery'**
>
> **`MFADeleteRE`** = **'<MfaDelete>([A-Za-z]+)</MfaDelete>'**
>
> **`VersionRE`** = **'<Status>([A-Za-z]+)</Status>'**
>
> **`VersioningBody`** = **'<?xml version="1.0" encoding="UTF-8"?>\n <VersioningConfiguration xmlns="http://s3.amazona**
>
> **`WebsiteBody`** = **'<?xml version="1.0" encoding="UTF-8"?>\n <WebsiteConfiguration xmlns="http://s3.amazonaws.con**
>
> **`WebsiteErrorFragment`** = **'<ErrorDocument><Key>%s</Key></ErrorDocument>'**

add_email_grant (*permission*, *email_address*, *recursive=False*, *headers=None*)
Convenience method that provides a quick way to add an email grant to a bucket. This method retrieves the current ACL, creates a new grant based on the parameters passed in, adds that grant to the ACL and then PUT's the new ACL back to S3.

Parameters

- **permission** (*string*) – The permission being granted. Should be one of: (READ, WRITE, READ_ACP, WRITE_ACP, FULL_CONTROL).
- **email_address** (*string*) – The email address associated with the AWS account your are granting the permission to.
- **recursive** (*boolean*) – A boolean value to controls whether the command will apply the grant to all keys within the bucket or not. The default value is False. By passing a True value, the call will iterate through all keys in the bucket and apply the same grant to each key. CAUTION: If you have a lot of keys, this could take a long time!

add_user_grant (*permission*, *user_id*, *recursive=False*, *headers=None*, *display_name=None*)
Convenience method that provides a quick way to add a canonical user grant to a bucket. This method retrieves the current ACL, creates a new grant based on the parameters passed in, adds that grant to the ACL and then PUT's the new ACL back to S3.

Parameters

- **permission** (*string*) – The permission being granted. Should be one of: (READ, WRITE, READ_ACP, WRITE_ACP, FULL_CONTROL).
- **user_id** (*string*) – The canonical user id associated with the AWS account your are granting the permission to.
- **recursive** (*boolean*) – A boolean value to controls whether the command will apply the grant to all keys within the bucket or not. The default value is False. By passing a True value, the call will iterate through all keys in the bucket and apply the same grant to each key. CAUTION: If you have a lot of keys, this could take a long time!
- **display_name** (*string*) – An option string containing the user's Display Name. Only required on Walrus.

cancel_multipart_upload (*key_name*, *upload_id*, *headers=None*)

complete_multipart_upload (*key_name*, *upload_id*, *xml_body*, *headers=None*)
Complete a multipart upload operation.

configure_lifecycle (*lifecycle_config*, *headers=None*)
Configure lifecycle for this bucket.

Parameters **lifecycle_config** (`boto.s3.lifecycle.Lifecycle`) – The lifecycle configuration you want to configure for this bucket.

configure_versioning (*versioning*, *mfa_delete=False*, *mfa_token=None*, *headers=None*)
Configure versioning for this bucket.

..note:: This feature is currently in beta.

Parameters

- **versioning** (*bool*) – A boolean indicating whether version is enabled (True) or disabled (False).
- **mfa_delete** (*bool*) – A boolean indicating whether the Multi-Factor Authentication Delete feature is enabled (True) or disabled (False). If mfa_delete is enabled then all Delete operations will require the token from your MFA device to be passed in the request.

- **mfa_token** (*tuple or list of strings*) – A tuple or list consisting of the serial number from the MFA device and the current value of the six-digit token associated with the device. This value is required when you are changing the status of the MfaDelete property of the bucket.

configure_website (*suffix*, *error_key=''*, *headers=None*)
Configure this bucket to act as a website

Parameters

- **suffix** (*str*) – Suffix that is appended to a request that is for a "directory" on the website endpoint (e.g. if the suffix is index.html and you make a request to samplebucket/images/ the data that is returned will be for the object with the key name images/index.html). The suffix must not be empty and must not include a slash character.
- **error_key** (*str*) – The object key name to use when a 4XX class error occurs. This is optional.

copy_key (*new_key_name*, *src_bucket_name*, *src_key_name*, *metadata=None*, *src_version_id=None*, *storage_class='STANDARD'*, *preserve_acl=False*, *encrypt_key=False*, *headers=None*, *query_args=None*)
Create a new key in the bucket by copying another existing key.

Parameters

- **new_key_name** (*string*) – The name of the new key
- **src_bucket_name** (*string*) – The name of the source bucket
- **src_key_name** (*string*) – The name of the source key
- **src_version_id** (*string*) – The version id for the key. This param is optional. If not specified, the newest version of the key will be copied.
- **metadata** (*dict*) – Metadata to be associated with new key. If metadata is supplied, it will replace the metadata of the source key being copied. If no metadata is supplied, the source key's metadata will be copied to the new key.
- **storage_class** (*string*) – The storage class of the new key. By default, the new key will use the standard storage class. Possible values are: STANDARD | REDUCED_REDUNDANCY
- **preserve_acl** (*bool*) – If True, the ACL from the source key will be copied to the destination key. If False, the destination key will have the default ACL. Note that preserving the ACL in the new key object will require two additional API calls to S3, one to retrieve the current ACL and one to set that ACL on the new object. If you don't care about the ACL, a value of False will be significantly more efficient.
- **encrypt_key** (*bool*) – If True, the new copy of the object will be encrypted on the server-side by S3 and will be stored in an encrypted form while at rest in S3.
- **headers** (*dict*) – A dictionary of header name/value pairs.
- **query_args** (*string*) – A string of additional querystring arguments to append to the request

Return type *boto.s3.key.Key* or subclass

Returns An instance of the newly created key object

delete (*headers=None*)

delete_key (*key_name*, *headers=None*, *version_id=None*, *mfa_token=None*)
Deletes a key from the bucket. If a version_id is provided, only that version of the key will be deleted.

Parameters

- **key_name** (*string*) – The key name to delete
- **version_id** (*string*) – The version ID (optional)
- **mfa_token** (*tuple or list of strings*) – A tuple or list consisting of the serial number from the MFA device and the current value of the six-digit token associated with the device. This value is required anytime you are deleting versioned objects from a bucket that has the MFADelete option on the bucket.

delete_keys (*keys*, *quiet=False*, *mfa_token=None*, *headers=None*)
Deletes a set of keys using S3's Multi-object delete API. If a VersionID is specified for that key then that version is removed. Returns a MultiDeleteResult Object, which contains Deleted and Error elements for each key you ask to delete.

Parameters

- **keys** (*list*) – A list of either key_names or (key_name, versionid) pairs or a list of Key instances.
- **quiet** (*boolean*) – In quiet mode the response includes only keys where the delete operation encountered an error. For a successful deletion, the operation does not return any information about the delete in the response body.
- **mfa_token** (*tuple or list of strings*) – A tuple or list consisting of the serial number from the MFA device and the current value of the six-digit token associated with the device. This value is required anytime you are deleting versioned objects from a bucket that has the MFADelete option on the bucket.

Returns An instance of MultiDeleteResult

delete_lifecycle_configuration (*headers=None*)
Removes all lifecycle configuration from the bucket.

delete_policy (*headers=None*)

delete_website_configuration (*headers=None*)
Removes all website configuration from the bucket.

disable_logging (*headers=None*)

enable_logging (*target_bucket*, *target_prefix='‘*, *headers=None*)

endElement (*name*, *value*, *connection*)

generate_url (*expires_in*, *method='GET'*, *headers=None*, *force_http=False*, *response_headers=None*, *expires_in_absolute=False*)

get_acl (*key_name='‘*, *headers=None*, *version_id=None*)

get_all_keys (*headers=None*, ***params*)
A lower-level method for listing contents of a bucket. This closely models the actual S3 API and requires you to manually handle the paging of results. For a higher-level method that handles the details of paging for you, you can use the list method.

Parameters

- **max_keys** (*int*) – The maximum number of keys to retrieve
- **prefix** (*string*) – The prefix of the keys you want to retrieve
- **marker** (*string*) – The “marker” of where you are in the result set

- **delimiter** (*string*) – If this optional, Unicode string parameter is included with your request, then keys that contain the same string between the prefix and the first occurrence of the delimiter will be rolled up into a single result element in the CommonPrefixes collection. These rolled-up keys are not returned elsewhere in the response.

Return type *ResultSet*

Returns The result from S3 listing the keys requested

get_all_multipart_uploads(*headers=None*, ***params*)
A lower-level, version-aware method for listing active MultiPart uploads for a bucket. This closely models the actual S3 API and requires you to manually handle the paging of results. For a higher-level method that handles the details of paging for you, you can use the list method.

Parameters

- **max_uploads** (*int*) – The maximum number of uploads to retrieve. Default value is 1000.
- **key_marker** (*string*) – Together with upload_id_marker, this parameter specifies the multipart upload after which listing should begin. If upload_id_marker is not specified, only the keys lexicographically greater than the specified key_marker will be included in the list.

  If upload_id_marker is specified, any multipart uploads for a key equal to the key_marker might also be included, provided those multipart uploads have upload IDs lexicographically greater than the specified upload_id_marker.
- **upload_id_marker** (*string*) – Together with key-marker, specifies the multipart upload after which listing should begin. If key_marker is not specified, the upload_id_marker parameter is ignored. Otherwise, any multipart uploads for a key equal to the key_marker might be included in the list only if they have an upload ID lexicographically greater than the specified upload_id_marker.

Return type *ResultSet*

Returns The result from S3 listing the uploads requested

get_all_versions(*headers=None*, ***params*)
A lower-level, version-aware method for listing contents of a bucket. This closely models the actual S3 API and requires you to manually handle the paging of results. For a higher-level method that handles the details of paging for you, you can use the list method.

Parameters

- **max_keys** (*int*) – The maximum number of keys to retrieve
- **prefix** (*string*) – The prefix of the keys you want to retrieve
- **key_marker** (*string*) – The "marker" of where you are in the result set with respect to keys.
- **version_id_marker** (*string*) – The "marker" of where you are in the result set with respect to version-id's.
- **delimiter** (*string*) – If this optional, Unicode string parameter is included with your request, then keys that contain the same string between the prefix and the first occurrence of the delimiter will be rolled up into a single result element in the CommonPrefixes collection. These rolled-up keys are not returned elsewhere in the response.

Return type *ResultSet*

Returns The result from S3 listing the keys requested

get_key(*key_name*, *headers=None*, *version_id=None*)
Check to see if a particular key exists within the bucket. This method uses a HEAD request to check for the existance of the key. Returns: An instance of a Key object or None

Parameters **key_name** (*string*) – The name of the key to retrieve

Return type *boto.s3.key.Key*

Returns A Key object from this bucket.

get_lifecycle_config(*headers=None*)
Returns the current lifecycle configuration on the bucket.

Return type `boto.s3.lifecycle.Lifecycle`

Returns A LifecycleConfig object that describes all current lifecycle rules in effect for the bucket.

get_location()
Returns the LocationConstraint for the bucket.

Return type str

Returns The LocationConstraint for the bucket or the empty string if no constraint was specified when bucket was created.

get_logging_status(*headers=None*)

get_policy(*headers=None*)
Returns the JSON policy associated with the bucket. The policy is returned as an uninterpreted JSON string.

get_request_payment(*headers=None*)

get_subresource(*subresource*, *key_name=''*, *headers=None*, *version_id=None*)
Get a subresource for a bucket or key.

Parameters

- **subresource** (*string*) – The subresource to get.
- **key_name** (*string*) – The key to operate on, or None to operate on the bucket.
- **headers** (`dict`) – Additional HTTP headers to include in the request.
- **src_version_id** (*string*) – Optional. The version id of the key to operate on. If not specified, operate on the newest version.

Return type string

Returns The value of the subresource.

get_versioning_status(*headers=None*)
Returns the current status of versioning on the bucket.

Return type *dict*

Returns A dictionary containing a key named 'Versioning' that can have a value of either Enabled, Disabled, or Suspended. Also, if MFADelete has ever been enabled on the bucket, the dictionary will contain a key named 'MFADelete' which will have a value of either Enabled or Suspended.

get_website_configuration(*headers=None*)
Returns the current status of website configuration on the bucket.

Return type *dict*

Returns A dictionary containing a Python representation of the XML response from S3. The overall structure is:

- WebsiteConfiguration
  - IndexDocument
    - Suffix : suffix that is appended to request that is for a "directory" on the website endpoint
    - ErrorDocument
      - Key : name of object to serve when an error occurs

get_website_endpoint()

Returns the fully qualified hostname to use is you want to access this bucket as a website. This doesn't validate whether the bucket has been correctly configured as a website or not.

get_xml_acl(*key_name=''*, *headers=None*, *version_id=None*)

initiate_multipart_upload(*key_name*, *headers=None*, *reduced_redundancy=False*, *metadata=None*, *encrypt_key=False*)

Start a multipart upload operation.

Parameters

- **key_name** (*string*) – The name of the key that will ultimately result from this multipart upload operation. This will be exactly as the key appears in the bucket after the upload process has been completed.
- **headers** (dict) – Additional HTTP headers to send and store with the resulting key in S3.
- **reduced_redundancy** (*boolean*) – In multipart uploads, the storage class is specified when initiating the upload, not when uploading individual parts. So if you want the resulting key to use the reduced redundancy storage class set this flag when you initiate the upload.
- **metadata** (dict) – Any metadata that you would like to set on the key that results from the multipart upload.
- **encrypt_key** (*bool*) – If True, the new copy of the object will be encrypted on the server-side by S3 and will be stored in an encrypted form while at rest in S3.

list(*prefix=''*, *delimiter=''*, *marker=''*, *headers=None*)

List key objects within a bucket. This returns an instance of an BucketListResultSet that automatically handles all of the result paging, etc. from S3. You just need to keep iterating until there are no more results.

Called with no arguments, this will return an iterator object across all keys within the bucket.

The Key objects returned by the iterator are obtained by parsing the results of a GET on the bucket, also known as the List Objects request. The XML returned by this request contains only a subset of the information about each key. Certain metadata fields such as Content-Type and user metadata are not available in the XML. Therefore, if you want these additional metadata fields you will have to do a HEAD request on the Key in the bucket.

Parameters

- **prefix** (*string*) – allows you to limit the listing to a particular prefix. For example, if you call the method with prefix='/foo/' then the iterator will only cycle through the keys that begin with the string '/foo/'.

- **delimiter** (*string*) – can be used in conjunction with the prefix to allow you to organize and browse your keys hierarchically. See: http://docs.amazonwebservices.com/AmazonS3/2006-03-01/ for more details.
- **marker** (*string*) – The "marker" of where you are in the result set

Return type *boto.s3.bucketlistresultset.BucketListResultSet*

Returns an instance of a BucketListResultSet that handles paging, etc

list_grants (*headers=None*)

list_multipart_uploads (*key_marker='', upload_id_marker='', headers=None*)
List multipart upload objects within a bucket. This returns an instance of an MultiPartUploadListResultSet that automatically handles all of the result paging, etc. from S3. You just need to keep iterating until there are no more results.

Parameters **marker** (*string*) – The "marker" of where you are in the result set

Return type *boto.s3.bucketlistresultset.BucketListResultSet*

Returns an instance of a BucketListResultSet that handles paging, etc

list_versions (*prefix='', delimiter='', key_marker='', version_id_marker='', headers=None*)
List version objects within a bucket. This returns an instance of an VersionedBucketListResultSet that automatically handles all of the result paging, etc. from S3. You just need to keep iterating until there are no more results. Called with no arguments, this will return an iterator object across all keys within the bucket.

Parameters

- **prefix** (*string*) – allows you to limit the listing to a particular prefix. For example, if you call the method with prefix='/foo/' then the iterator will only cycle through the keys that begin with the string '/foo/'.
- **delimiter** (*string*) – can be used in conjunction with the prefix to allow you to organize and browse your keys hierarchically. See: http://docs.amazonwebservices.com/AmazonS3/2006-03-01/ for more details.
- **marker** (*string*) – The "marker" of where you are in the result set

Return type *boto.s3.bucketlistresultset.BucketListResultSet*

Returns an instance of a BucketListResultSet that handles paging, etc

lookup (*key_name, headers=None*)
Deprecated: Please use get_key method.

Parameters **key_name** (*string*) – The name of the key to retrieve

Return type *boto.s3.key.Key*

Returns A Key object from this bucket.

make_public (*recursive=False, headers=None*)

new_key (*key_name=None*)
Creates a new key

Parameters **key_name** (*string*) – The name of the key to create

Return type *boto.s3.key.Key* or subclass

Returns An instance of the newly created key object

set_acl (*acl_or_str, key_name='', headers=None, version_id=None*)

set_as_logging_target (*headers=None*)

set_canned_acl (*acl_str*, *key_name=''*, *headers=None*, *version_id=None*)

set_key_class (*key_class*)
Set the Key class associated with this bucket. By default, this would be the boto.s3.key.Key class but if you want to subclass that for some reason this allows you to associate your new class with a bucket so that when you call bucket.new_key() or when you get a listing of keys in the bucket you will get an instances of your key class rather than the default.

Parameters **key_class** (*class*) – A subclass of Key that can be more specific

set_policy (*policy*, *headers=None*)
Add or replace the JSON policy associated with the bucket.

Parameters **policy** (*str*) – The JSON policy as a string.

set_request_payment (*payer='BucketOwner'*, *headers=None*)

set_subresource (*subresource*, *value*, *key_name=''*, *headers=None*, *version_id=None*)
Set a subresource for a bucket or key.

Parameters

- **subresource** (*string*) – The subresource to set.
- **value** (*string*) – The value of the subresource.
- **key_name** (*string*) – The key to operate on, or None to operate on the bucket.
- **headers** (dict) – Additional HTTP headers to include in the request.
- **src_version_id** (*string*) – Optional. The version id of the key to operate on. If not specified, operate on the newest version.

set_xml_acl (*acl_str*, *key_name=''*, *headers=None*, *version_id=None*, *query_args='acl'*)

startElement (*name*, *attrs*, *connection*)

class boto.s3.bucket.**S3WebsiteEndpointTranslate**

trans_region = defaultdict(<function <lambda>>, {'ap-northeast-1': 's3-website-ap-northeast-1', 'sa-east-1': 's3-webs

classmethod **translate_region** (*reg*)

boto.s3.bucketlistresultset

class boto.s3.bucketlistresultset.**BucketListResultSet** (*bucket=None*, *prefix=''*, *delimiter=''*, *marker=''*, *headers=None*)
A resultset for listing keys within a bucket. Uses the bucket_lister generator function and implements the iterator interface. This transparently handles the results paging from S3 so even if you have many thousands of keys within the bucket you can iterate over all keys in a reasonably efficient manner.

class boto.s3.bucketlistresultset.**MultiPartUploadListResultSet** (*bucket=None*, *key_marker=''*, *upload_id_marker=''*, *headers=None*)
A resultset for listing multipart uploads within a bucket. Uses the multipart_upload_lister generator function and implements the iterator interface. This transparently handles the results paging from S3 so even if you have many thousands of uploads within the bucket you can iterate over all keys in a reasonably efficient manner.

class `boto.s3.bucketlistresultset.`**`VersionedBucketListResultSet`**(*bucket=None*, *prefix=''*, *delimiter=''*, *key_marker=''*, *version_id_marker=''*, *headers=None*)

A resultset for listing versions within a bucket. Uses the bucket_lister generator function and implements the iterator interface. This transparently handles the results paging from S3 so even if you have many thousands of keys within the bucket you can iterate over all keys in a reasonably efficient manner.

`boto.s3.bucketlistresultset.`**`bucket_lister`**(*bucket*, *prefix=''*, *delimiter=''*, *marker=''*, *headers=None*)

A generator function for listing keys in a bucket.

`boto.s3.bucketlistresultset.`**`multipart_upload_lister`**(*bucket*, *key_marker=''*, *upload_id_marker=''*, *headers=None*)

A generator function for listing multipart uploads in a bucket.

`boto.s3.bucketlistresultset.`**`versioned_bucket_lister`**(*bucket*, *prefix=''*, *delimiter=''*, *key_marker=''*, *version_id_marker=''*, *headers=None*)

A generator function for listing versions in a bucket.

boto.s3.connection

class `boto.s3.connection.`**`Location`**

`APNortheast` = **'ap-northeast-1'**

`APSoutheast` = **'ap-southeast-1'**

`DEFAULT` = **''**

`EU` = **'EU'**

`SAEast` = **'sa-east-1'**

`USWest` = **'us-west-1'**

class `boto.s3.connection.`**`OrdinaryCallingFormat`**

`build_path_base`(*bucket*, *key=''*)

`get_bucket_server`(*server*, *bucket*)

class `boto.s3.connection.`**`ProtocolIndependentOrdinaryCallingFormat`**

`build_url_base`(*connection*, *protocol*, *server*, *bucket*, *key=''*)

class `boto.s3.connection.`**`S3Connection`**(*aws_access_key_id=None, aws_secret_access_key=None, is_secure=True, port=None, proxy=None, proxy_port=None, proxy_user=None, proxy_pass=None, host='s3.amazonaws.com', debug=0, https_connection_factory=None, calling_format=<boto.s3.connection.SubdomainCallingFormat object>, path='/', provider='aws', bucket_class=<class 'boto.s3.bucket.Bucket'>, security_token=None, suppress_consec_slashes=True, anon=False*)

`DefaultHost` = **'s3.amazonaws.com'**

`QueryString` = **'Signature=%s&Expires=%d&AWSAccessKeyId=%s'**

`build_post_form_args`(*bucket_name, key, expires_in=6000, acl=None, success_action_redirect=None, max_content_length=None, http_method='http', fields=None, conditions=None*)

Taken from the AWS book Python examples and modified for use with boto This only returns the arguments required for the post form, not the actual form. This does not return the file input field which also needs to be added

Parameters

- **`bucket_name`** (*string*) – Bucket to submit to
- **`key`** (*string*) – Key name, optionally add ${filename} to the end to attach the submitted filename
- **`expires_in`** (*integer*) – Time (in seconds) before this expires, defaults to 6000
- **`acl`** (*boto.s3.acl.ACL*) – ACL rule to use, if any
- **`success_action_redirect`** (*string*) – URL to redirect to on success
- **`max_content_length`** (*integer*) – Maximum size for this file
- **`http_method`** (*string*) – HTTP Method to use, "http" or "https"

Return type *dict*

Returns

A dictionary containing field names/values as well as a url to POST to

```
{
    "action": action_url_to_post_to,
    "fields": [
        {
            "name": field_name,
            "value":  field_value
        },
        {
            "name": field_name2,
            "value": field_value2
        }
    ]
}
```

`build_post_policy`(*expiration_time, conditions*)

Taken from the AWS book Python examples and modified for use with boto

create_bucket (*bucket_name*, *headers=None*, *location=''*, *policy=None*)
Creates a new located bucket. By default it's in the USA. You can pass Location.EU to create an European bucket.

Parameters

- **bucket_name** (*string*) – The name of the new bucket
- **headers** (dict) – Additional headers to pass along with the request to AWS.
- **location** (*boto.s3.connection.Location*) – The location of the new bucket
- **policy** (boto.s3.acl.CannedACLStrings) – A canned ACL policy that will be applied to the new key in S3.

delete_bucket (*bucket*, *headers=None*)

generate_url (*expires_in*, *method*, *bucket=''*, *key=''*, *headers=None*, *query_auth=True*, *force_http=False*, *response_headers=None*, *expires_in_absolute=False*)

get_all_buckets (*headers=None*)

get_bucket (*bucket_name*, *validate=True*, *headers=None*)

get_canonical_user_id (*headers=None*)
Convenience method that returns the "CanonicalUserID" of the user who's credentials are associated with the connection. The only way to get this value is to do a GET request on the service which returns all buckets associated with the account. As part of that response, the canonical userid is returned. This method simply does all of that and then returns just the user id.

Return type string

Returns A string containing the canonical user id.

lookup (*bucket_name*, *validate=True*, *headers=None*)

make_request (*method*, *bucket=''*, *key=''*, *headers=None*, *data=''*, *query_args=None*, *sender=None*, *override_num_retries=None*)

set_bucket_class (*bucket_class*)
Set the Bucket class associated with this bucket. By default, this would be the boto.s3.key.Bucket class but if you want to subclass that for some reason this allows you to associate your new class.

Parameters **bucket_class** (*class*) – A subclass of Bucket that can be more specific

class boto.s3.connection.**SubdomainCallingFormat**

get_bucket_server (**args*, ***kwargs*)

class boto.s3.connection.**VHostCallingFormat**

get_bucket_server (**args*, ***kwargs*)

boto.s3.connection.**assert_case_insensitive** (*f*)

boto.s3.connection.**check_lowercase_bucketname** (*n*)
Bucket names must not contain uppercase characters. We check for this by appending a lowercase character and testing with islower(). Note this also covers cases like numeric bucket names with dashes.

```
>>> check_lowercase_bucketname("Aaaa")
Traceback (most recent call last):
...
BotoClientError: S3Error: Bucket names cannot contain upper-case
```

```
characters when using either the sub-domain or virtual hosting calling
format.
```

```
>>> check_lowercase_bucketname("1234-5678-9123")
True
>>> check_lowercase_bucketname("abcdefg1234")
True
```

boto.s3.key

class `boto.s3.key.`**`Key`**(*bucket=None*, *name=None*)

`BufferSize` **= 8192**

`DefaultContentType` **= 'application/octet-stream'**

`add_email_grant`(*permission*, *email_address*, *headers=None*)

Convenience method that provides a quick way to add an email grant to a key. This method retrieves the current ACL, creates a new grant based on the parameters passed in, adds that grant to the ACL and then PUT's the new ACL back to S3.

Parameters

- **`permission`** (*`string`*) – The permission being granted. Should be one of: (READ, WRITE, READ_ACP, WRITE_ACP, FULL_CONTROL).
- **`email_address`** (*`string`*) – The email address associated with the AWS account your are granting the permission to.
- **`recursive`** (*`boolean`*) – A boolean value to controls whether the command will apply the grant to all keys within the bucket or not. The default value is False. By passing a True value, the call will iterate through all keys in the bucket and apply the same grant to each key. CAUTION: If you have a lot of keys, this could take a long time!

`add_user_grant`(*permission*, *user_id*, *headers=None*, *display_name=None*)

Convenience method that provides a quick way to add a canonical user grant to a key. This method retrieves the current ACL, creates a new grant based on the parameters passed in, adds that grant to the ACL and then PUT's the new ACL back to S3.

Parameters

- **`permission`** (*`string`*) – The permission being granted. Should be one of: (READ, WRITE, READ_ACP, WRITE_ACP, FULL_CONTROL).
- **`user_id`** (*`string`*) – The canonical user id associated with the AWS account your are granting the permission to.
- **`display_name`** (*`string`*) – An option string containing the user's Display Name. Only required on Walrus.

`change_storage_class`(*new_storage_class*, *dst_bucket=None*)

Change the storage class of an existing key. Depending on whether a different destination bucket is supplied or not, this will either move the item within the bucket, preserving all metadata and ACL info bucket changing the storage class or it will copy the item to the provided destination bucket, also preserving metadata and ACL info.

Parameters

- **new_storage_class** (*string*) – The new storage class for the Key. Possible values are: * STANDARD * REDUCED_REDUNDANCY
- **dst_bucket** (*string*) – The name of a destination bucket. If not provided the current bucket of the key will be used.

close()

closed = False

compute_md5(*fp*, *size=None*)

Parameters

- **fp** (file) – File pointer to the file to MD5 hash. The file pointer will be reset to the same position before the method returns.
- **size** (*int*) – (optional) The Maximum number of bytes to read from the file pointer (fp). This is useful when uploading a file in multiple parts where the file is being split inplace into different parts. Less bytes may be available.

Return type tuple

Returns A tuple containing the hex digest version of the MD5 hash as the first element and the base64 encoded version of the plain digest as the second element.

copy(*dst_bucket*, *dst_key*, *metadata=None*, *reduced_redundancy=False*, *preserve_acl=False*, *encrypt_key=False*)

Copy this Key to another bucket.

Parameters

- **dst_bucket** (*string*) – The name of the destination bucket
- **dst_key** (*string*) – The name of the destination key
- **metadata** (dict) – Metadata to be associated with new key. If metadata is supplied, it will replace the metadata of the source key being copied. If no metadata is supplied, the source key's metadata will be copied to the new key.
- **reduced_redundancy** (*bool*) – If True, this will force the storage class of the new Key to be REDUCED_REDUNDANCY regardless of the storage class of the key being copied. The Reduced Redundancy Storage (RRS) feature of S3, provides lower redundancy at lower storage cost.
- **preserve_acl** (*bool*) – If True, the ACL from the source key will be copied to the destination key. If False, the destination key will have the default ACL. Note that preserving the ACL in the new key object will require two additional API calls to S3, one to retrieve the current ACL and one to set that ACL on the new object. If you don't care about the ACL, a value of False will be significantly more efficient.
- **encrypt_key** (*bool*) – If True, the new copy of the object will be encrypted on the server-side by S3 and will be stored in an encrypted form while at rest in S3.

Return type *boto.s3.key.Key* or subclass

Returns An instance of the newly created key object

delete()

Delete this key from S3

endElement(*name*, *value*, *connection*)

exists()

Returns True if the key exists

Return type bool

Returns Whether the key exists on S3

generate_url(*expires_in*, *method='GET'*, *headers=None*, *query_auth=True*, *force_http=False*, *response_headers=None*, *expires_in_absolute=False*)
Generate a URL to access this key.

Parameters

- **expires_in** (*int*) – How long the url is valid for, in seconds
- **method** (*string*) – The method to use for retrieving the file (default is GET)
- **headers** (dict) – Any headers to pass along in the request
- **query_auth** (*bool*) –

Return type string

Returns The URL to access the key

get_acl(*headers=None*)

get_contents_as_string(*headers=None*, *cb=None*, *num_cb=10*, *torrent=False*, *version_id=None*, *response_headers=None*)
Retrieve an object from S3 using the name of the Key object as the key in S3. Return the contents of the object as a string. See get_contents_to_file method for details about the parameters.

Parameters

- **headers** (dict) – Any additional headers to send in the request
- **cb** (*int*) – a callback function that will be called to report progress on the upload. The callback should accept two integer parameters, the first representing the number of bytes that have been successfully transmitted to S3 and the second representing the size of the to be transmitted object.
- **num_cb** – (optional) If a callback is specified with the cb parameter this parameter determines the granularity of the callback by defining the maximum number of times the callback will be called during the file transfer.
- **torrent** (*bool*) – If True, returns the contents of a torrent file as a string.
- **response_headers** (dict) – A dictionary containing HTTP headers/values that will override any headers associated with the stored object in the response. See http://goo.gl/EWOPb for details.

Return type string

Returns The contents of the file as a string

get_contents_to_file(*fp*, *headers=None*, *cb=None*, *num_cb=10*, *torrent=False*, *version_id=None*, *res_download_handler=None*, *response_headers=None*)
Retrieve an object from S3 using the name of the Key object as the key in S3. Write the contents of the object to the file pointed to by 'fp'.

Parameters

- **fp** (*File -like object*) –
- **headers** (dict) – additional HTTP headers that will be sent with the GET request.
- **cb** (*int*) – a callback function that will be called to report progress on the upload. The callback should accept two integer parameters, the first representing the number of bytes

that have been successfully transmitted to S3 and the second representing the size of the to be transmitted object.

- **num_cb** – (optional) If a callback is specified with the cb parameter this parameter determines the granularity of the callback by defining the maximum number of times the callback will be called during the file transfer.
- **torrent** (*bool*) – If True, returns the contents of a torrent file as a string.
- **res_download_handler** – If provided, this handler will perform the download.
- **response_headers** (dict) – A dictionary containing HTTP headers/values that will override any headers associated with the stored object in the response. See http://goo.gl/EWOPb for details.

get_contents_to_filename (*filename, headers=None, cb=None, num_cb=10, torrent=False, version_id=None, res_download_handler=None, response_headers=None*)

Retrieve an object from S3 using the name of the Key object as the key in S3. Store contents of the object to a file named by 'filename'. See get_contents_to_file method for details about the parameters.

Parameters

- **filename** (*string*) – The filename of where to put the file contents
- **headers** (dict) – Any additional headers to send in the request
- **cb** (*int*) – a callback function that will be called to report progress on the upload. The callback should accept two integer parameters, the first representing the number of bytes that have been successfully transmitted to S3 and the second representing the size of the to be transmitted object.
- **num_cb** – (optional) If a callback is specified with the cb parameter this parameter determines the granularity of the callback by defining the maximum number of times the callback will be called during the file transfer.
- **torrent** (*bool*) – If True, returns the contents of a torrent file as a string.
- **res_download_handler** – If provided, this handler will perform the download.
- **response_headers** (dict) – A dictionary containing HTTP headers/values that will override any headers associated with the stored object in the response. See http://goo.gl/EWOPb for details.

get_file (*fp, headers=None, cb=None, num_cb=10, torrent=False, version_id=None, override_num_retries=None, response_headers=None*)

Retrieves a file from an S3 Key

Parameters

- **fp** (file) – File pointer to put the data into
- **cb** (*int*) – a callback function that will be called to report progress on the upload. The callback should accept two integer parameters, the first representing the number of bytes that have been successfully transmitted to S3 and the second representing the size of the to be transmitted object.
- **num_cb** – (optional) If a callback is specified with the cb parameter this parameter determines the granularity of the callback by defining the maximum number of times the callback will be called during the file transfer.
- **torrent** (*bool*) – Flag for whether to get a torrent for the file

- **override_num_retries** (*int*) – If not None will override configured num_retries parameter for underlying GET.
- **response_headers** (dict) – A dictionary containing HTTP headers/values that will override any headers associated with the stored object in the response. See http://goo.gl/EWOPb for details.

Param headers to send when retrieving the files

get_md5_from_hexdigest (*md5_hexdigest*)
A utility function to create the 2-tuple (md5hexdigest, base64md5) from just having a precalculated md5_hexdigest.

get_metadata (*name*)

get_torrent_file (*fp*, *headers=None*, *cb=None*, *num_cb=10*)
Get a torrent file (see to get_file)

Parameters

- **fp** (file) – The file pointer of where to put the torrent
- **headers** (dict) – Headers to be passed
- **cb** (*int*) – a callback function that will be called to report progress on the upload. The callback should accept two integer parameters, the first representing the number of bytes that have been successfully transmitted to S3 and the second representing the size of the to be transmitted object.
- **num_cb** – (optional) If a callback is specified with the cb parameter this parameter determines the granularity of the callback by defining the maximum number of times the callback will be called during the file transfer.

get_xml_acl (*headers=None*)

handle_encryption_headers (*resp*)

handle_version_headers (*resp*, *force=False*)

make_public (*headers=None*)

next ()
By providing a next method, the key object supports use as an iterator. For example, you can now say:

for bytes in key: write bytes to a file or whatever

All of the HTTP connection stuff is handled for you.

open (*mode='r'*, *headers=None*, *query_args=None*, *override_num_retries=None*)

open_read (*headers=None*, *query_args='‘*, *override_num_retries=None*, *response_headers=None*)
Open this key for reading

Parameters

- **headers** (dict) – Headers to pass in the web request
- **query_args** (*string*) – Arguments to pass in the query string (ie, 'torrent')
- **override_num_retries** (*int*) – If not None will override configured num_retries parameter for underlying GET.
- **response_headers** (dict) – A dictionary containing HTTP headers/values that will override any headers associated with the stored object in the response. See http://goo.gl/EWOPb for details.

open_write(*headers=None*, *override_num_retries=None*)
Open this key for writing. Not yet implemented

Parameters

- **headers** (dict) – Headers to pass in the write request
- **override_num_retries** (*int*) – If not None will override configured num_retries parameter for underlying PUT.

provider

read(*size=0*)

send_file(*fp*, *headers=None*, *cb=None*, *num_cb=10*, *query_args=None*, *chunked_transfer=False*, *size=None*)
Upload a file to a key into a bucket on S3.

Parameters

- **fp** (file) – The file pointer to upload. The file pointer must point point at the offset from which you wish to upload. ie. if uploading the full file, it should point at the start of the file. Normally when a file is opened for reading, the fp will point at the first byte. See the bytes parameter below for more info.
- **headers** (dict) – The headers to pass along with the PUT request
- **cb** (*function*) – a callback function that will be called to report progress on the upload. The callback should accept two integer parameters, the first representing the number of bytes that have been successfully transmitted to S3 and the second representing the size of the to be transmitted object.
- **num_cb** (*int*) – (optional) If a callback is specified with the cb parameter this parameter determines the granularity of the callback by defining the maximum number of times the callback will be called during the file transfer. Providing a negative integer will cause your callback to be called with each buffer read.
- **size** (*int*) – (optional) The Maximum number of bytes to read from the file pointer (fp). This is useful when uploading a file in multiple parts where you are splitting the file up into different ranges to be uploaded. If not specified, the default behaviour is to read all bytes from the file pointer. Less bytes may be available.

set_acl(*acl_str*, *headers=None*)

set_canned_acl(*acl_str*, *headers=None*)

set_contents_from_file(*fp*, *headers=None*, *replace=True*, *cb=None*, *num_cb=10*, *policy=None*, *md5=None*, *reduced_redundancy=False*, *query_args=None*, *encrypt_key=False*, *size=None*)
Store an object in S3 using the name of the Key object as the key in S3 and the contents of the file pointed to by 'fp' as the contents.

Parameters

- **fp** (file) – the file whose contents to upload
- **headers** (dict) – Additional HTTP headers that will be sent with the PUT request.
- **replace** (*bool*) – If this parameter is False, the method will first check to see if an object exists in the bucket with the same key. If it does, it won't overwrite it. The default value is True which will overwrite the object.
- **cb** (*int*) – a callback function that will be called to report progress on the upload. The callback should accept two integer parameters, the first representing the number of bytes

that have been successfully transmitted to S3 and the second representing the size of the to be transmitted object.

- **num_cb** – (optional) If a callback is specified with the cb parameter this parameter determines the granularity of the callback by defining the maximum number of times the callback will be called during the file transfer.
- **policy** (`boto.s3.acl.CannedACLStrings`) – A canned ACL policy that will be applied to the new key in S3.
- **md5** (*A tuple containing the hexdigest version of the MD5 checksum of the file as the first element and the Base64-encoded version of the plain checksum as the second element. This is the same format returned by the compute_md5 method.*) – If you need to compute the MD5 for any reason prior to upload, it's silly to have to do it twice so this param, if present, will be used as the MD5 values of the file. Otherwise, the checksum will be computed.
- **reduced_redundancy** (*bool*) – If True, this will set the storage class of the new Key to be REDUCED_REDUNDANCY. The Reduced Redundancy Storage (RRS) feature of S3, provides lower redundancy at lower storage cost.
- **encrypt_key** (*bool*) – If True, the new copy of the object will be encrypted on the server-side by S3 and will be stored in an encrypted form while at rest in S3.
- **size** (*int*) – (optional) The Maximum number of bytes to read from the file pointer (fp). This is useful when uploading a file in multiple parts where you are splitting the file up into different ranges to be uploaded. If not specified, the default behaviour is to read all bytes from the file pointer. Less bytes may be available.

set_contents_from_filename(*filename*, *headers=None*, *replace=True*, *cb=None*, *num_cb=10*, *policy=None*, *md5=None*, *reduced_redundancy=False*, *encrypt_key=False*)

Store an object in S3 using the name of the Key object as the key in S3 and the contents of the file named by 'filename'. See set_contents_from_file method for details about the parameters.

Parameters

- **filename** (*string*) – The name of the file that you want to put onto S3
- **headers** (`dict`) – Additional headers to pass along with the request to AWS.
- **replace** (*bool*) – If True, replaces the contents of the file if it already exists.
- **cb** (*int*) – a callback function that will be called to report progress on the upload. The callback should accept two integer parameters, the first representing the number of bytes that have been successfully transmitted to S3 and the second representing the size of the to be transmitted object.
- **num_cb** – (optional) If a callback is specified with the cb parameter this parameter determines the granularity of the callback by defining the maximum number of times the callback will be called during the file transfer.
- **policy** (`boto.s3.acl.CannedACLStrings`) – A canned ACL policy that will be applied to the new key in S3.
- **md5** (*A tuple containing the hexdigest version of the MD5 checksum of the file as the first element and the Base64-encoded version of the plain checksum as the second element. This is the same format returned by the compute_md5 method.*) – If you need to compute the MD5 for any reason prior to

upload, it's silly to have to do it twice so this param, if present, will be used as the MD5 values of the file. Otherwise, the checksum will be computed.

- **reduced_redundancy** (*bool*) – If True, this will set the storage class of the new Key to be REDUCED_REDUNDANCY. The Reduced Redundancy Storage (RRS) feature of S3, provides lower redundancy at lower storage cost.
- **encrypt_key** (*bool*) – If True, the new copy of the object will be encrypted on the server-side by S3 and will be stored in an encrypted form while at rest in S3.

set_contents_from_stream(*fp*, *headers=None*, *replace=True*, *cb=None*, *num_cb=10*, *policy=None*, *reduced_redundancy=False*, *query_args=None*, *size=None*)

Store an object using the name of the Key object as the key in cloud and the contents of the data stream pointed to by 'fp' as the contents. The stream object is not seekable and total size is not known. This has the implication that we can't specify the Content-Size and Content-MD5 in the header. So for huge uploads, the delay in calculating MD5 is avoided but with a penalty of inability to verify the integrity of the uploaded data.

Parameters

- **fp** (file) – the file whose contents are to be uploaded
- **headers** (dict) – additional HTTP headers to be sent with the PUT request.
- **replace** (*bool*) – If this parameter is False, the method will first check to see if an object exists in the bucket with the same key. If it does, it won't overwrite it. The default value is True which will overwrite the object.
- **cb** (*function*) – a callback function that will be called to report progress on the upload. The callback should accept two integer parameters, the first representing the number of bytes that have been successfully transmitted to GS and the second representing the total number of bytes that need to be transmitted.
- **num_cb** (*int*) – (optional) If a callback is specified with the cb parameter, this parameter determines the granularity of the callback by defining the maximum number of times the callback will be called during the file transfer.
- **policy** (boto.gs.acl.CannedACLStrings) – A canned ACL policy that will be applied to the new key in GS.
- **reduced_redundancy** (*bool*) – If True, this will set the storage class of the new Key to be REDUCED_REDUNDANCY. The Reduced Redundancy Storage (RRS) feature of S3, provides lower redundancy at lower storage cost.
- **size** (*int*) – (optional) The Maximum number of bytes to read from the file pointer (fp). This is useful when uploading a file in multiple parts where you are splitting the file up into different ranges to be uploaded. If not specified, the default behaviour is to read all bytes from the file pointer. Less bytes may be available.

set_contents_from_string(*s*, *headers=None*, *replace=True*, *cb=None*, *num_cb=10*, *policy=None*, *md5=None*, *reduced_redundancy=False*, *encrypt_key=False*)

Store an object in S3 using the name of the Key object as the key in S3 and the string 's' as the contents. See set_contents_from_file method for details about the parameters.

Parameters

- **headers** (dict) – Additional headers to pass along with the request to AWS.
- **replace** (*bool*) – If True, replaces the contents of the file if it already exists.

- **cb** (*int*) – a callback function that will be called to report progress on the upload. The callback should accept two integer parameters, the first representing the number of bytes that have been successfully transmitted to S3 and the second representing the size of the to be transmitted object.
- **num_cb** – (optional) If a callback is specified with the cb parameter this parameter determines the granularity of the callback by defining the maximum number of times the callback will be called during the file transfer.
- **policy** (`boto.s3.acl.CannedACLStrings`) – A canned ACL policy that will be applied to the new key in S3.
- **md5** (*A tuple containing the hexdigest version of the MD5 checksum of the file as the first element and the Base64-encoded version of the plain checksum as the second element. This is the same format returned by the compute_md5 method.*) – If you need to compute the MD5 for any reason prior to upload, it's silly to have to do it twice so this param, if present, will be used as the MD5 values of the file. Otherwise, the checksum will be computed.
- **reduced_redundancy** (*bool*) – If True, this will set the storage class of the new Key to be REDUCED_REDUNDANCY. The Reduced Redundancy Storage (RRS) feature of S3, provides lower redundancy at lower storage cost.
- **encrypt_key** (*bool*) – If True, the new copy of the object will be encrypted on the server-side by S3 and will be stored in an encrypted form while at rest in S3.

set_metadata (*name*, *value*)

set_xml_acl (*acl_str*, *headers=None*)

startElement (*name*, *attrs*, *connection*)

update_metadata (*d*)

boto.s3.prefix

class `boto.s3.prefix.`**Prefix** (*bucket=None*, *name=None*)

endElement (*name*, *value*, *connection*)

startElement (*name*, *attrs*, *connection*)

boto.s3.user

class `boto.s3.user.`**User** (*parent=None*, *id=''*, *display_name=''*)

endElement (*name*, *value*, *connection*)

startElement (*name*, *attrs*, *connection*)

to_xml (*element_name='Owner'*)

boto.s3.multipart

class `boto.s3.multipart.`**`CompleteMultiPartUpload`**(*bucket=None*)

Represents a completed MultiPart Upload. Contains the following useful attributes:

- location - The URI of the completed upload
- **bucket_name - The name of the bucket in which the upload** is contained
- key_name - The name of the new, completed key
- etag - The MD5 hash of the completed, combined upload

`endElement`(*name, value, connection*)

`startElement`(*name, attrs, connection*)

class `boto.s3.multipart.`**`MultiPartUpload`**(*bucket=None*)

Represents a MultiPart Upload operation.

`cancel_upload`()

Cancels a MultiPart Upload operation. The storage consumed by any previously uploaded parts will be freed. However, if any part uploads are currently in progress, those part uploads might or might not succeed. As a result, it might be necessary to abort a given multipart upload multiple times in order to completely free all storage consumed by all parts.

`complete_upload`()

Complete the MultiPart Upload operation. This method should be called when all parts of the file have been successfully uploaded to S3.

Return type `boto.s3.multipart.CompletedMultiPartUpload`

Returns An object representing the completed upload.

`copy_part_from_key`(*src_bucket_name, src_key_name, part_num, start=None, end=None*)

Copy another part of this MultiPart Upload.

Parameters

- **`src_bucket_name`** (*`string`*) – Name of the bucket containing the source key
- **`src_key_name`** (*`string`*) – Name of the source key
- **`part_num`** (*`int`*) – The number of this part.
- **`start`** (*`int`*) – Zero-based byte offset to start copying from
- **`end`** (*`int`*) – Zero-based byte offset to copy to

`endElement`(*name, value, connection*)

`get_all_parts`(*max_parts=None, part_number_marker=None*)

Return the uploaded parts of this MultiPart Upload. This is a lower-level method that requires you to manually page through results. To simplify this process, you can just use the object itself as an iterator and it will automatically handle all of the paging with S3.

`startElement`(*name, attrs, connection*)

`to_xml`()

`upload_part_from_file`(*fp, part_num, headers=None, replace=True, cb=None, num_cb=10, policy=None, md5=None, size=None*)

Upload another part of this MultiPart Upload.

Parameters

- **`fp`** (`file`) – The file object you want to upload.

- **part_num** (*int*) – The number of this part.

The other parameters are exactly as defined for the `boto.s3.key.Key` set_contents_from_file method.

class `boto.s3.multipart.`**`Part`**(*bucket=None*)

Represents a single part in a MultiPart upload. Attributes include:

•part_number - The integer part number

•last_modified - The last modified date of this part

•etag - The MD5 hash of this part

•size - The size, in bytes, of this part

`endElement`(*name*, *value*, *connection*)

`startElement`(*name*, *attrs*, *connection*)

`boto.s3.multipart.`**`part_lister`**(*mpupload*, *part_number_marker=None*)

A generator function for listing parts of a multipart upload.

boto.s3.resumable_download_handler

class `boto.s3.resumable_download_handler.`**`ByteTranslatingCallbackHandler`**(*proxied_cb*, *download_start_point*)

Proxy class that translates progress callbacks made by boto.s3.Key.get_file(), taking into account that we're resuming a download.

`call`(*total_bytes_uploaded*, *total_size*)

class `boto.s3.resumable_download_handler.`**`ResumableDownloadHandler`**(*tracker_file_name=None*, *num_retries=None*)

Handler for resumable downloads.

Constructor. Instantiate once for each downloaded file.

Parameters

- **tracker_file_name** (*string*) – optional file name to save tracking info about this download. If supplied and the current process fails the download, it can be retried in a new process. If called with an existing file containing an unexpired timestamp, we'll resume the transfer for this file; else we'll start a new resumable download.
- **num_retries** (*int*) – the number of times we'll re-try a resumable download making no progress. (Count resets every time we get progress, so download can span many more than this number of retries.)

`ETAG_REGEX` = **'([a-z0-9]{32})\n'**

`RETRYABLE_EXCEPTIONS` = **(<class 'httplib.HTTPException'>, <type 'exceptions.IOError'>, <class 'socket.error'>, <cl**

`get_file`(*key*, *fp*, *headers*, *cb=None*, *num_cb=10*, *torrent=False*, *version_id=None*)

Retrieves a file from a Key :type key: `boto.s3.key.Key` or subclass :param key: The Key object from which upload is to be downloaded

Parameters

- **fp** (`file`) – File pointer into which data should be downloaded
- **cb** (*function*) – (optional) a callback function that will be called to report progress on the download. The callback should accept two integer parameters, the first representing

the number of bytes that have been successfully transmitted from the storage service and the second representing the total number of bytes that need to be transmitted.

- **num_cb** (*int*) – (optional) If a callback is specified with the cb parameter this parameter determines the granularity of the callback by defining the maximum number of times the callback will be called during the file transfer.
- **torrent** (*bool*) – Flag for whether to get a torrent for the file
- **version_id** (*string*) – The version ID (optional)

Param headers to send when retrieving the files

Raises ResumableDownloadException if a problem occurs during the transfer.

`boto.s3.resumable_download_handler.`**`get_cur_file_size`**(*fp*, *position_to_eof=False*)
Returns size of file, optionally leaving fp positioned at EOF.

boto.s3.deletemarker

class `boto.s3.deletemarker.`**`DeleteMarker`**(*bucket=None*, *name=None*)

`endElement`(*name*, *value*, *connection*)

`startElement`(*name*, *attrs*, *connection*)

mturk

boto.mturk

boto.mturk.connection

class `boto.mturk.connection.`**`Assignment`**(*connection*)
Class to extract an Assignment structure from a response (used in ResultSet)

Will have attributes named as per the Developer Guide, e.g. AssignmentId, WorkerId, HITId, Answer, etc

`endElement`(*name*, *value*, *connection*)

class `boto.mturk.connection.`**`BaseAutoResultElement`**(*connection*)
Base class to automatically add attributes when parsing XML

`endElement`(*name*, *value*, *connection*)

`startElement`(*name*, *attrs*, *connection*)

class `boto.mturk.connection.`**`HIT`**(*connection*)
Class to extract a HIT structure from a response (used in ResultSet)

Will have attributes named as per the Developer Guide, e.g. HITId, HITTypeId, CreationTime

`expired`
Has this HIT expired yet?

class `boto.mturk.connection.`**`MTurkConnection`**(*aws_access_key_id=None*, *aws_secret_access_key=None*, *is_secure=True*, *port=None*, *proxy=None*, *proxy_port=None*, *proxy_user=None*, *proxy_pass=None*, *host=None*, *debug=0*, *https_connection_factory=None*)

`APIVersion` = **'2008-08-02'**

`approve_assignment`(*assignment_id*, *feedback=None*)

`assign_qualification`(*qualification_type_id*, *worker_id*, *value=1*, *send_notification=True*)

`block_worker`(*worker_id*, *reason*)
Block a worker from working on my tasks.

`change_hit_type_of_hit`(*hit_id*, *hit_type*)
Change the HIT type of an existing HIT. Note that the reward associated with the new HIT type must match the reward of the current HIT type in order for the operation to be valid.

`create_hit`(*hit_type=None*, *question=None*, *lifetime=datetime.timedelta(7)*, *max_assignments=1*, *title=None*, *description=None*, *keywords=None*, *reward=None*, *duration=datetime.timedelta(7)*, *approval_delay=None*, *annotation=None*, *questions=None*, *qualifications=None*, *response_groups=None*)
Creates a new HIT. Returns a ResultSet See: http://docs.amazonwebservices.com/AWSMechanicalTurkRequester/2006-10-31/ApiReference_CreateHITOperation.html

`create_qualification_type`(*name*, *description*, *status*, *keywords=None*, *retry_delay=None*, *test=None*, *answer_key=None*, *answer_key_xml=None*, *test_duration=None*, *auto_granted=False*, *auto_granted_value=1*)
Create a new Qualification Type.

name: This will be visible to workers and must be unique for a given requester.

description: description shown to workers. Max 2000 characters.

status: 'Active' or 'Inactive'

keywords: list of keyword strings or comma separated string. Max length of 1000 characters when concatenated with commas.

retry_delay: number of seconds after requesting a qualification the worker must wait before they can ask again. If not specified, workers can only request this qualification once.

test: a QuestionForm

answer_key: an XML string of your answer key, for automatically scored qualification tests. (Consider implementing an AnswerKey class for this to support.)

test_duration: the number of seconds a worker has to complete the test.

auto_granted: if True, requests for the Qualification are granted immediately. Can't coexist with a test.

auto_granted_value: auto_granted qualifications are given this value.

`disable_hit`(*hit_id*, *response_groups=None*)
Remove a HIT from the Mechanical Turk marketplace, approves all submitted assignments that have not already been approved or rejected, and disposes of the HIT and all assignment data.

Assignments for the HIT that have already been submitted, but not yet approved or rejected, will be automatically approved. Assignments in progress at the time of the call to DisableHIT will be approved once the assignments are submitted. You will be charged for approval of these assignments. DisableHIT completely disposes of the HIT and all submitted assignment data. Assignment results data cannot be retrieved for a HIT that has been disposed.

It is not possible to re-enable a HIT once it has been disabled. To make the work from a disabled HIT available again, create a new HIT.

`dispose_hit`(*hit_id*)

Dispose of a HIT that is no longer needed.

Only HITs in the "reviewable" state, with all submitted assignments approved or rejected, can be disposed. A Requester can call GetReviewableHITs to determine which HITs are reviewable, then call GetAssignmentsForHIT to retrieve the assignments. Disposing of a HIT removes the HIT from the results of a call to GetReviewableHITs.

`dispose_qualification_type`(*qualification_type_id*)

TODO: Document.

static `duration_as_seconds`(*duration*)

`expire_hit`(*hit_id*)

Expire a HIT that is no longer needed.

The effect is identical to the HIT expiring on its own. The HIT no longer appears on the Mechanical Turk web site, and no new Workers are allowed to accept the HIT. Workers who have accepted the HIT prior to expiration are allowed to complete it or return it, or allow the assignment duration to elapse (abandon the HIT). Once all remaining assignments have been submitted, the expired HIT becomes"reviewable", and will be returned by a call to GetReviewableHITs.

`extend_hit`(*hit_id*, *assignments_increment=None*, *expiration_increment=None*)

Increase the maximum number of assignments, or extend the expiration date, of an existing HIT.

NOTE: If a HIT has a status of Reviewable and the HIT is extended to make it Available, the HIT will not be returned by GetReviewableHITs, and its submitted assignments will not be returned by GetAssignmentsForHIT, until the HIT is Reviewable again. Assignment auto-approval will still happen on its original schedule, even if the HIT has been extended. Be sure to retrieve and approve (or reject) submitted assignments before extending the HIT, if so desired.

`get_account_balance`()

`get_all_hits`()

Return all of a Requester's HITs

Despite what search_hits says, it does not return all hits, but instead returns a page of hits. This method will pull the hits from the server 100 at a time, but will yield the results iteratively, so subsequent requests are made on demand.

`get_assignments`(*hit_id*, *status=None*, *sort_by='SubmitTime'*, *sort_direction='Ascending'*, *page_size=10*, *page_number=1*, *response_groups=None*)

Retrieves completed assignments for a HIT. Use this operation to retrieve the results for a HIT.

The returned ResultSet will have the following attributes:

NumResults The number of assignments on the page in the filtered results list, equivalent to the number of assignments being returned by this call. A non-negative integer

PageNumber The number of the page in the filtered results list being returned. A positive integer

TotalNumResults The total number of HITs in the filtered results list based on this call. A non-negative integer

The ResultSet will contain zero or more Assignment objects

get_help (*about*, *help_type='Operation'*)
Return information about the Mechanical Turk Service operations and response group NOTE - this is basically useless as it just returns the URL of the documentation

help_type: either 'Operation' or 'ResponseGroup'

get_hit (*hit_id*, *response_groups=None*)

static **get_keywords_as_string** (*keywords*)
Returns a comma+space-separated string of keywords from either a list or a string

static **get_price_as_price** (*reward*)
Returns a Price data structure from either a float or a Price

get_qualification_requests (*qualification_type_id*, *sort_by='Expiration'*, *sort_direction='Ascending'*, *page_size=10*, *page_number=1*)
TODO: Document.

get_qualification_score (*qualification_type_id*, *worker_id*)
TODO: Document.

get_qualification_type (*qualification_type_id*)

get_qualifications_for_qualification_type (*qualification_type_id*)

get_reviewable_hits (*hit_type=None*, *status='Reviewable'*, *sort_by='Expiration'*, *sort_direction='Ascending'*, *page_size=10*, *page_number=1*)
Retrieve the HITs that have a status of Reviewable, or HITs that have a status of Reviewing, and that belong to the Requester calling the operation.

grant_bonus (*worker_id*, *assignment_id*, *bonus_price*, *reason*)
Issues a payment of money from your account to a Worker. To be eligible for a bonus, the Worker must have submitted results for one of your HITs, and have had those results approved or rejected. This payment happens separately from the reward you pay to the Worker when you approve the Worker's assignment. The Bonus must be passed in as an instance of the Price object.

grant_qualification (*qualification_request_id*, *integer_value=1*)
TODO: Document.

notify_workers (*worker_ids*, *subject*, *message_text*)
Send a text message to workers.

register_hit_type (*title*, *description*, *reward*, *duration*, *keywords=None*, *approval_delay=None*, *qual_req=None*)
Register a new HIT Type title, description are strings reward is a Price object duration can be a timedelta, or an object castable to an int

reject_assignment (*assignment_id*, *feedback=None*)

revoke_qualification (*subject_id*, *qualification_type_id*, *reason=None*)
TODO: Document.

search_hits (*sort_by='CreationTime'*, *sort_direction='Ascending'*, *page_size=10*, *page_number=1*, *response_groups=None*)
Return a page of a Requester's HITs, on behalf of the Requester. The operation returns HITs of any status, except for HITs that have been disposed with the DisposeHIT operation. Note: The SearchHITs operation does not accept any search parameters that filter the results.

search_qualification_types (*query=None*, *sort_by='Name'*, *sort_direction='Ascending'*, *page_size=10*, *page_number=1*, *must_be_requestable=True*, *must_be_owned_by_caller=True*)
TODO: Document.

set_email_notification(*hit_type*, *email*, *event_types=None*)
Performs a SetHITTypeNotification operation to set email notification for a specified HIT type

set_rest_notification(*hit_type*, *url*, *event_types=None*)
Performs a SetHITTypeNotification operation to set REST notification for a specified HIT type

set_reviewing(*hit_id*, *revert=None*)
Update a HIT with a status of Reviewable to have a status of Reviewing, or reverts a Reviewing HIT back to the Reviewable status.

Only HITs with a status of Reviewable can be updated with a status of Reviewing. Similarly, only Reviewing HITs can be reverted back to a status of Reviewable.

unblock_worker(*worker_id*, *reason*)
Unblock a worker from working on my tasks.

update_qualification_score(*qualification_type_id*, *worker_id*, *value*)
TODO: Document.

update_qualification_type(*qualification_type_id*, *description=None*, *status=None*, *retry_delay=None*, *test=None*, *answer_key=None*, *test_duration=None*, *auto_granted=None*, *auto_granted_value=None*)

exception boto.mturk.connection.**MTurkRequestError**(*status*, *reason*, *body=None*)
Error for MTurk Requests

class boto.mturk.connection.**Qualification**(*connection*)
Class to extract an Qualification structure from a response (used in ResultSet)

Will have attributes named as per the Developer Guide such as QualificationTypeId, IntegerValue. Does not seem to contain GrantTime.

class boto.mturk.connection.**QualificationRequest**(*connection*)
Class to extract an QualificationRequest structure from a response (used in ResultSet)

Will have attributes named as per the Developer Guide, e.g. QualificationRequestId, QualificationTypeId, SubjectId, etc

TODO: Ensure that Test and Answer attribute are treated properly if the qualification requires a test. These attributes are XML-encoded.

class boto.mturk.connection.**QualificationType**(*connection*)
Class to extract an QualificationType structure from a response (used in ResultSet)

Will have attributes named as per the Developer Guide, e.g. QualificationTypeId, CreationTime, Name, etc

class boto.mturk.connection.**QuestionFormAnswer**(*connection*)
Class to extract Answers from inside the embedded XML QuestionFormAnswers element inside the Answer element which is part of the Assignment structure

A QuestionFormAnswers element contains an Answer element for each question in the HIT or Qualification test for which the Worker provided an answer. Each Answer contains a QuestionIdentifier element whose value corresponds to the QuestionIdentifier of a Question in the QuestionForm. See the QuestionForm data structure for more information about questions and answer specifications.

If the question expects a free-text answer, the Answer element contains a FreeText element. This element contains the Worker's answer

NOTE - currently really only supports free-text and selection answers

endElement(*name*, *value*, *connection*)

boto.mturk.notification

Provides NotificationMessage and Event classes, with utility methods, for implementations of the Mechanical Turk Notification API.

class `boto.mturk.notification.`**`Event`**(*d*)

class `boto.mturk.notification.`**`NotificationMessage`**(*d*)

Constructor; expects parameter d to be a dict of string parameters from a REST transport notification message

`EVENT_PATTERN` **= 'Event\\.(?P<n>\\d+)\\.(?P<param>\\w+)'**

`EVENT_RE` **= <_sre.SRE_Pattern object>**

`NOTIFICATION_VERSION` **= '2006-05-05'**

`NOTIFICATION_WSDL` **= 'http://mechanicalturk.amazonaws.com/AWSMechanicalTurk/2006-05-05/AWSMechanicalTurl**

`OPERATION_NAME` **= 'Notify'**

`SERVICE_NAME` **= 'AWSMechanicalTurkRequesterNotification'**

`verify`(*secret_key*)

Verifies the authenticity of a notification message.

TODO: This is doing a form of authentication and this functionality should really be merged with the pluggable authentication mechanism at some point.

boto.mturk.price

class `boto.mturk.price.`**`Price`**(*amount=0.0*, *currency_code='USD'*)

`endElement`(*name*, *value*, *connection*)

`get_as_params`(*label*, *ord=1*)

`startElement`(*name*, *attrs*, *connection*)

boto.mturk.qualification

class `boto.mturk.qualification.`**`AdultRequirement`**(*comparator*, *integer_value*, *required_to_preview=False*)

Requires workers to acknowledge that they are over 18 and that they agree to work on potentially offensive content. The value type is boolean, 1 (required), 0 (not required, the default).

class `boto.mturk.qualification.`**`LocaleRequirement`**(*comparator*, *locale*, *required_to_preview=False*)

A Qualification requirement based on the Worker's location. The Worker's location is specified by the Worker to Mechanical Turk when the Worker creates his account.

`get_as_params`()

class `boto.mturk.qualification.`**`NumberHitsApprovedRequirement`**(*comparator*, *integer_value*, *required_to_preview=False*)

Specifies the total number of HITs submitted by a Worker that have been approved. The value is an integer greater than or equal to 0.

class boto.mturk.qualification.**PercentAssignmentsAbandonedRequirement**(*comparator*, *integer_value*, *required_to_preview=False*)

The percentage of assignments the Worker has abandoned (allowed the deadline to elapse), over all assignments the Worker has accepted. The value is an integer between 0 and 100.

class boto.mturk.qualification.**PercentAssignmentsApprovedRequirement**(*comparator*, *integer_value*, *required_to_preview=False*)

The percentage of assignments the Worker has submitted that were subsequently approved by the Requester, over all assignments the Worker has submitted. The value is an integer between 0 and 100.

class boto.mturk.qualification.**PercentAssignmentsRejectedRequirement**(*comparator*, *integer_value*, *required_to_preview=False*)

The percentage of assignments the Worker has submitted that were subsequently rejected by the Requester, over all assignments the Worker has submitted. The value is an integer between 0 and 100.

class boto.mturk.qualification.**PercentAssignmentsReturnedRequirement**(*comparator*, *integer_value*, *required_to_preview=False*)

The percentage of assignments the Worker has returned, over all assignments the Worker has accepted. The value is an integer between 0 and 100.

class boto.mturk.qualification.**PercentAssignmentsSubmittedRequirement**(*comparator*, *integer_value*, *required_to_preview=False*)

The percentage of assignments the Worker has submitted, over all assignments the Worker has accepted. The value is an integer between 0 and 100.

class boto.mturk.qualification.**Qualifications**(*requirements=None*)

add(*req*)

get_as_params()

class boto.mturk.qualification.**Requirement**(*qualification_type_id*, *comparator*, *integer_value=None*, *required_to_preview=False*)

Representation of a single requirement

get_as_params()

boto.mturk.question

class boto.mturk.question.**AnswerSpecification**(*spec*)

get_as_xml()

template = '<AnswerSpecification>%(spec)s</AnswerSpecification>'

class boto.mturk.question.**Application**(*width*, *height*, ***parameters*)

get_as_xml()

get_inner_content(*content*)

parameter_template = '<Name>%(name)s</Name><Value>%(value)s</Value>'

template = '<Application><%(class_)s>%(content)s</%(class_)s></Application>'

class boto.mturk.question.**Binary**(*type*, *subtype*, *url*, *alt_text*)

template = '<Binary><MimeType><Type>%(type)s</Type><SubType>%(subtype)s</SubType></MimeType><DataUl

class boto.mturk.question.**Constraint**

get_as_xml()

get_attributes()

class boto.mturk.question.**Constraints**

get_as_xml()

template = '<Constraints>%(content)s</Constraints>'

class boto.mturk.question.**ExternalQuestion**(*external_url*, *frame_height*)
An object for constructing an External Question.

get_as_params(*label='ExternalQuestion'*)

get_as_xml()

schema_url = 'http://mechanicalturk.amazonaws.com/AWSMechanicalTurkDataSchemas/2006-07-14/ExternalQuestion

template = '<ExternalQuestion xmlns="http://mechanicalturk.amazonaws.com/AWSMechanicalTurkDataSchemas/200

class boto.mturk.question.**FileUploadAnswer**(*min_bytes*, *max_bytes*)

get_as_xml()

template = '<FileUploadAnswer><MinFileSizeInBytes>%(min_bytes)d</MinFileSizeInBytes><MaxFileSizeInBytes>%

class boto.mturk.question.**Flash**(*url*, **args*, ***kwargs*)

get_inner_content(*content*)

class boto.mturk.question.**FormattedContent**(*content*)

schema_url = 'http://mechanicalturk.amazonaws.com/AWSMechanicalTurkDataSchemas/2006-07-14/FormattedConte

template = '<FormattedContent><![CDATA[%(content)s]]></FormattedContent>'

class boto.mturk.question.**FreeTextAnswer**(*default=None*, *constraints=None*, *num_lines=None*)

get_as_xml()

template = '<FreeTextAnswer>%(items)s</FreeTextAnswer>'

class `boto.mturk.question.`**`JavaApplet`**(*path*, *filename*, **args*, ***kwargs*)

`get_inner_content`(*content*)

class `boto.mturk.question.`**`LengthConstraint`**(*min_length=None*, *max_length=None*)

`attribute_names` **= ('minLength', 'maxLength')**

`template` **= '<Length %(attrs)s />'**

class `boto.mturk.question.`**`List`**
A bulleted list suitable for OrderedContent or Overview content

`get_as_xml`()

class `boto.mturk.question.`**`NumberOfLinesSuggestion`**(*num_lines=1*)

`get_as_xml`()

`template` **= '<NumberOfLinesSuggestion>%(num_lines)s</NumberOfLinesSuggestion>'**

class `boto.mturk.question.`**`NumericConstraint`**(*min_value=None*, *max_value=None*)

`attribute_names` **= ('minValue', 'maxValue')**

`template` **= '<IsNumeric %(attrs)s />'**

class `boto.mturk.question.`**`OrderedContent`**

`append_field`(*field*, *value*)

`get_as_xml`()

class `boto.mturk.question.`**`Overview`**

`get_as_params`(*label='Overview'*)

`get_as_xml`()

`template` **= '<Overview>%(content)s</Overview>'**

class `boto.mturk.question.`**`Question`**(*identifier*, *content*, *answer_spec*, *is_required=False*, *display_name=None*)

`get_as_params`(*label='Question'*)

`get_as_xml`()

`template` **= '<Question>%(items)s</Question>'**

class `boto.mturk.question.`**`QuestionContent`**

`get_as_xml`()

`template` **= '<QuestionContent>%(content)s</QuestionContent>'**

class `boto.mturk.question.`**`QuestionForm`**
From the AMT API docs:

The top-most element of the QuestionForm data structure is a QuestionForm element. This element contains optional Overview elements and one or more Question elements. There can be any number of these two element types listed in any order. The following example structure has an Overview element and a Question element followed by a second Overview element and Question element–all within the same QuestionForm.

```
<QuestionForm xmlns="[the QuestionForm schema URL]">
    <Overview>
        [...]
    </Overview>
    <Question>
        [...]
    </Question>
    <Overview>
        [...]
    </Overview>
    <Question>
        [...]
    </Question>
    [...]
</QuestionForm>
```

QuestionForm is implemented as a list, so to construct a QuestionForm, simply append Questions and Overviews (with at least one Question).

get_as_xml()

is_valid()

schema_url = 'http://mechanicalturk.amazonaws.com/AWSMechanicalTurkDataSchemas/2005-10-01/QuestionForm.xs

xml_template = '<QuestionForm xmlns="http://mechanicalturk.amazonaws.com/AWSMechanicalTurkDataSchemas/2

class boto.mturk.question.**RegExConstraint**(*pattern*, *error_text=None*, *flags=None*)

attribute_names = ('regex', 'errorText', 'flags')

template = '<AnswerFormatRegex %(attrs)s />'

class boto.mturk.question.**SelectionAnswer**(*min=1*, *max=1*, *style=None*, *selections=None*, *type='text'*, *other=False*)

A class to generate SelectionAnswer XML data structures. Does not yet implement Binary selection options.

ACCEPTED_STYLES = ['radiobutton', 'dropdown', 'checkbox', 'list', 'combobox', 'multichooser']

MAX_SELECTION_COUNT_XML_TEMPLATE = '<MaxSelectionCount>%s</MaxSelectionCount>'

MIN_SELECTION_COUNT_XML_TEMPLATE = '<MinSelectionCount>%s</MinSelectionCount>'

OTHER_SELECTION_ELEMENT_NAME = 'OtherSelection'

SELECTIONANSWER_XML_TEMPLATE = '<SelectionAnswer>%s%s<Selections>%s</Selections></SelectionAnswer>'

SELECTION_VALUE_XML_TEMPLATE = '<%s>%s</%s>'

SELECTION_XML_TEMPLATE = '<Selection><SelectionIdentifier>%s</SelectionIdentifier>%s</Selection>'

STYLE_XML_TEMPLATE = '<StyleSuggestion>%s</StyleSuggestion>'

get_as_xml()

class boto.mturk.question.**SimpleField**(*field*, *value*)

A Simple name/value pair that can be easily rendered as XML.

```
>>> SimpleField('Text', 'A text string').get_as_xml()
'<Text>A text string</Text>'
```

template = '<%(field)s>%(value)s</%(field)s>'

class `boto.mturk.question.`**`ValidatingXML`**

`validate``()`

class `boto.mturk.question.`**`XMLTemplate`**

`get_as_xml``()`

Boto Config

Introduction

There is a growing list of configuration options for the boto library. Many of these options can be passed into the constructors for top-level objects such as connections. Some options, such as credentials, can also be read from environment variables (e.g. `AWS_ACCESS_KEY_ID` and `AWS_SECRET_ACCESS_KEY`). But there is no central place to manage these options. So, the development version of boto has now introduced the notion of boto config files.

Details

A boto config file is simply a .ini format configuration file that specifies values for options that control the behavior of the boto library. Upon startup, the boto library looks for configuration files in the following locations and in the following order:

- /etc/boto.cfg - for site-wide settings that all users on this machine will use
- ~/.boto - for user-specific settings

The options are merged into a single, in-memory configuration that is available as `boto.config`. The *`boto.pyami.config.Config`* class is a subclass of the standard Python `ConfigParser.SafeConfigParser` object and inherits all of the methods of that object. In addition, the boto *`Config`* class defines additional methods that are described on the PyamiConfigMethods page.

Sections

The following sections and options are currently recognized within the boto config file.

Credentials

The Credentials section is used to specify the AWS credentials used for all boto requests. The order of precedence for authentication credentials is:

- Credentials passed into Connection class constructor.
- Credentials specified by environment variables
- Credentials specified as options in the config file.

This section defines the following options: `aws_access_key_id` and `aws_secret_access_key`. The former being your aws key id and the latter being the secret key.

For example:

```
[Credentials]
aws_access_key_id = <your access key>
aws_secret_access_key = <your secret key>
```

Please notice that quote characters are not used to either side of the '=' operator even when both your aws access key id and secret key are strings.

Boto

The Boto section is used to specify options that control the operaton of boto itself. This section defines the following options:

debug Controls the level of debug messages that will be printed by the boto library. The following values are defined:

```
0 - no debug messages are printed
1 - basic debug messages from boto are printed
2 - all boto debugging messages plus request/response messages from
↪httplib
```

proxy The name of the proxy host to use for connecting to AWS.

proxy_port The port number to use to connect to the proxy host.

proxy_user The user name to use when authenticating with proxy host.

proxy_pass The password to use when authenticating with proxy host.

num_retries The number of times to retry failed requests to an AWS server. If boto receives an error from AWS, it will attempt to recover and retry the request. The default number of retries is 5 but you can change the default with this option.

As an example:

```
[Boto]
debug = 0
num_retries = 10

proxy = myproxy.com
proxy_port = 8080
proxy_user = foo
proxy_pass = bar
```

Precedence

Even if you have your boto config setup, you can also have credentials and options stored in environmental variables or you can explicitly pass them to method calls i.e.:

```
>>> boto.connect_ec2('<KEY_ID>','<SECRET_KEY>')
```

In these cases where these options can be found in more than one place boto will first use the explicitly supplied arguments, if none found it will then look for them amidst environment variables and if that fails it will use the ones in boto config.

API Reference

boto

boto

class `boto.`**`NullHandler`**(*level=0*)

Initializes the instance - basically setting the formatter to None and the filter list to empty.

`emit`(*record*)

`boto.`**`check_extensions`**(*module_name*, *module_path*)

This function checks for extensions to boto modules. It should be called in the __init__.py file of all boto modules. See: http://code.google.com/p/boto/wiki/ExtendModules

for details.

`boto.`**`connect_autoscale`**(*aws_access_key_id=None*, *aws_secret_access_key=None*, ***kwargs*)

Parameters

- **`aws_access_key_id`** (*`string`*) – Your AWS Access Key ID
- **`aws_secret_access_key`** (*`string`*) – Your AWS Secret Access Key

Return type *`boto.ec2.autoscale.AutoScaleConnection`*

Returns A connection to Amazon's Auto Scaling Service

`boto.`**`connect_cloudformation`**(*aws_access_key_id=None*, *aws_secret_access_key=None*, ***kwargs*)

Parameters

- **`aws_access_key_id`** (*`string`*) – Your AWS Access Key ID
- **`aws_secret_access_key`** (*`string`*) – Your AWS Secret Access Key

Return type `boto.cloudformation.CloudFormationConnection`

Returns A connection to Amazon's CloudFormation Service

`boto.`**`connect_cloudfront`**(*aws_access_key_id=None*, *aws_secret_access_key=None*, ***kwargs*)

Parameters

- **`aws_access_key_id`** (*`string`*) – Your AWS Access Key ID
- **`aws_secret_access_key`** (*`string`*) – Your AWS Secret Access Key

Return type *`boto.fps.connection.FPSConnection`*

Returns A connection to FPS

`boto.`**`connect_cloudwatch`**(*aws_access_key_id=None*, *aws_secret_access_key=None*, ***kwargs*)

Parameters

- **`aws_access_key_id`** (*`string`*) – Your AWS Access Key ID
- **`aws_secret_access_key`** (*`string`*) – Your AWS Secret Access Key

Return type *`boto.ec2.cloudwatch.CloudWatchConnection`*

Returns A connection to Amazon's EC2 Monitoring service

`boto.`**`connect_dynamodb`**(*aws_access_key_id=None*, *aws_secret_access_key=None*, ***kwargs*)

Parameters

- **aws_access_key_id** (*string*) – Your AWS Access Key ID
- **aws_secret_access_key** (*string*) – Your AWS Secret Access Key

Return type *boto.dynamodb.layer2.Layer2*

Returns A connection to the Layer2 interface for DynamoDB.

boto.**connect_ec2**(*aws_access_key_id=None*, *aws_secret_access_key=None*, ***kwargs*)

Parameters

- **aws_access_key_id** (*string*) – Your AWS Access Key ID
- **aws_secret_access_key** (*string*) – Your AWS Secret Access Key

Return type *boto.ec2.connection.EC2Connection*

Returns A connection to Amazon's EC2

boto.**connect_ec2_endpoint**(*url*, *aws_access_key_id=None*, *aws_secret_access_key=None*, ***kwargs*)

Connect to an EC2 Api endpoint. Additional arguments are passed through to connect_ec2.

Parameters

- **url** (*string*) – A url for the ec2 api endpoint to connect to
- **aws_access_key_id** (*string*) – Your AWS Access Key ID
- **aws_secret_access_key** (*string*) – Your AWS Secret Access Key

Return type *boto.ec2.connection.EC2Connection*

Returns A connection to Eucalyptus server

boto.**connect_elb**(*aws_access_key_id=None*, *aws_secret_access_key=None*, ***kwargs*)

Parameters

- **aws_access_key_id** (*string*) – Your AWS Access Key ID
- **aws_secret_access_key** (*string*) – Your AWS Secret Access Key

Return type *boto.ec2.elb.ELBConnection*

Returns A connection to Amazon's Load Balancing Service

boto.**connect_emr**(*aws_access_key_id=None*, *aws_secret_access_key=None*, ***kwargs*)

Parameters

- **aws_access_key_id** (*string*) – Your AWS Access Key ID
- **aws_secret_access_key** (*string*) – Your AWS Secret Access Key

Return type boto.emr.EmrConnection

Returns A connection to Elastic mapreduce

boto.**connect_euca**(*host=None*, *aws_access_key_id=None*, *aws_secret_access_key=None*, *port=8773*, *path='/services/Eucalyptus'*, *is_secure=False*, ***kwargs*)

Connect to a Eucalyptus service.

Parameters

- **host** (*string*) – the host name or ip address of the Eucalyptus server
- **aws_access_key_id** (*string*) – Your AWS Access Key ID

- **aws_secret_access_key** (*string*) – Your AWS Secret Access Key

Return type *boto.ec2.connection.EC2Connection*

Returns A connection to Eucalyptus server

boto.**connect_fps**(*aws_access_key_id=None*, *aws_secret_access_key=None*, ***kwargs*)

Parameters

- **aws_access_key_id** (*string*) – Your AWS Access Key ID
- **aws_secret_access_key** (*string*) – Your AWS Secret Access Key

Return type *boto.fps.connection.FPSConnection*

Returns A connection to FPS

boto.**connect_gs**(*gs_access_key_id=None*, *gs_secret_access_key=None*, ***kwargs*)

@type gs_access_key_id: string @param gs_access_key_id: Your Google Cloud Storage Access Key ID

@type gs_secret_access_key: string @param gs_secret_access_key: Your Google Cloud Storage Secret Access Key

@rtype: L{GSConnection<boto.gs.connection.GSConnection>} @return: A connection to Google's Storage service

boto.**connect_ia**(*ia_access_key_id=None*, *ia_secret_access_key=None*, *is_secure=False*, ***kwargs*)

Connect to the Internet Archive via their S3-like API.

Parameters

- **ia_access_key_id** (*string*) – Your IA Access Key ID. This will also look in your boto config file for an entry in the Credentials section called "ia_access_key_id"
- **ia_secret_access_key** (*string*) – Your IA Secret Access Key. This will also look in your boto config file for an entry in the Credentials section called "ia_secret_access_key"

Return type *boto.s3.connection.S3Connection*

Returns A connection to the Internet Archive

boto.**connect_iam**(*aws_access_key_id=None*, *aws_secret_access_key=None*, ***kwargs*)

Parameters

- **aws_access_key_id** (*string*) – Your AWS Access Key ID
- **aws_secret_access_key** (*string*) – Your AWS Secret Access Key

Return type boto.iam.IAMConnection

Returns A connection to Amazon's IAM

boto.**connect_mturk**(*aws_access_key_id=None*, *aws_secret_access_key=None*, ***kwargs*)

Parameters

- **aws_access_key_id** (*string*) – Your AWS Access Key ID
- **aws_secret_access_key** (*string*) – Your AWS Secret Access Key

Return type *boto.mturk.connection.MTurkConnection*

Returns A connection to MTurk

boto.**connect_rds**(*aws_access_key_id=None*, *aws_secret_access_key=None*, ***kwargs*)

Parameters

- **aws_access_key_id** (*string*) – Your AWS Access Key ID
- **aws_secret_access_key** (*string*) – Your AWS Secret Access Key

Return type *boto.rds.RDSConnection*

Returns A connection to RDS

boto.**connect_route53**(*aws_access_key_id=None*, *aws_secret_access_key=None*, ***kwargs*)

Parameters

- **aws_access_key_id** (*string*) – Your AWS Access Key ID
- **aws_secret_access_key** (*string*) – Your AWS Secret Access Key

Return type boto.dns.Route53Connection

Returns A connection to Amazon's Route53 DNS Service

boto.**connect_s3**(*aws_access_key_id=None*, *aws_secret_access_key=None*, ***kwargs*)

Parameters

- **aws_access_key_id** (*string*) – Your AWS Access Key ID
- **aws_secret_access_key** (*string*) – Your AWS Secret Access Key

Return type *boto.s3.connection.S3Connection*

Returns A connection to Amazon's S3

boto.**connect_sdb**(*aws_access_key_id=None*, *aws_secret_access_key=None*, ***kwargs*)

Parameters

- **aws_access_key_id** (*string*) – Your AWS Access Key ID
- **aws_secret_access_key** (*string*) – Your AWS Secret Access Key

Return type *boto.sdb.connection.SDBConnection*

Returns A connection to Amazon's SDB

boto.**connect_ses**(*aws_access_key_id=None*, *aws_secret_access_key=None*, ***kwargs*)

Parameters

- **aws_access_key_id** (*string*) – Your AWS Access Key ID
- **aws_secret_access_key** (*string*) – Your AWS Secret Access Key

Return type boto.ses.SESConnection

Returns A connection to Amazon's SES

boto.**connect_sns**(*aws_access_key_id=None*, *aws_secret_access_key=None*, ***kwargs*)

Parameters

- **aws_access_key_id** (*string*) – Your AWS Access Key ID
- **aws_secret_access_key** (*string*) – Your AWS Secret Access Key

Return type *boto.sns.SNSConnection*

Returns A connection to Amazon's SNS

boto.**connect_sqs**(*aws_access_key_id=None*, *aws_secret_access_key=None*, ***kwargs*)

Parameters

- **aws_access_key_id** (*string*) – Your AWS Access Key ID
- **aws_secret_access_key** (*string*) – Your AWS Secret Access Key

Return type *boto.sqs.connection.SQSConnection*

Returns A connection to Amazon's SQS

boto.**connect_sts**(*aws_access_key_id=None*, *aws_secret_access_key=None*, ***kwargs*)

Parameters

- **aws_access_key_id** (*string*) – Your AWS Access Key ID
- **aws_secret_access_key** (*string*) – Your AWS Secret Access Key

Return type *boto.sts.STSConnection*

Returns A connection to Amazon's STS

boto.**connect_vpc**(*aws_access_key_id=None*, *aws_secret_access_key=None*, ***kwargs*)

Parameters

- **aws_access_key_id** (*string*) – Your AWS Access Key ID
- **aws_secret_access_key** (*string*) – Your AWS Secret Access Key

Return type *boto.vpc.VPCConnection*

Returns A connection to VPC

boto.**connect_walrus**(*host=None*, *aws_access_key_id=None*, *aws_secret_access_key=None*, *port=8773*, *path='/services/Walrus'*, *is_secure=False*, ***kwargs*)
Connect to a Walrus service.

Parameters

- **host** (*string*) – the host name or ip address of the Walrus server
- **aws_access_key_id** (*string*) – Your AWS Access Key ID
- **aws_secret_access_key** (*string*) – Your AWS Secret Access Key

Return type *boto.s3.connection.S3Connection*

Returns A connection to Walrus

boto.**init_logging**()

boto.**lookup**(*service*, *name*)

boto.**set_file_logger**(*name*, *filepath*, *level=20*, *format_string=None*)

boto.**set_stream_logger**(*name*, *level=10*, *format_string=None*)

boto.**storage_uri**(*uri_str*, *default_scheme='file'*, *debug=0*, *validate=True*, *bucket_storage_uri_class=<class 'boto.storage_uri.BucketStorageUri'>*, *suppress_consec_slashes=True*)
Instantiate a StorageUri from a URI string.

Parameters

- **uri_str** (*string*) – URI naming bucket + optional object.
- **default_scheme** (*string*) – default scheme for scheme-less URIs.
- **debug** (*int*) – debug level to pass in to boto connection (range 0..2).
- **validate** (*bool*) – whether to check for bucket name validity.

- **`bucket_storage_uri_class`** (*`BucketStorageUri interface.`*) – Allows mocking for unit tests.
- **`suppress_consec_slashes`** – If provided, controls whether consecutive slashes will be suppressed in key paths.

We allow validate to be disabled to allow caller to implement bucket-level wildcarding (outside the boto library; see gsutil).

Return type `boto.StorageUri` subclass

Returns StorageUri subclass for given URI.

`uri_str` must be one of the following formats:

- gs://bucket/name
- s3://bucket/name
- gs://bucket
- s3://bucket
- filename

The last example uses the default scheme ('file', unless overridden)

`boto.`**`storage_uri_for_key`**(*key*)

Returns a StorageUri for the given key.

Parameters **`key`** (*`boto.s3.key.Key`* or subclass) – URI naming bucket + optional object.

boto.connection

Handles basic connections to AWS

class `boto.connection.`**`AWSAuthConnection`**(*host*, *aws_access_key_id=None*, *aws_secret_access_key=None*, *is_secure=True*, *port=None*, *proxy=None*, *proxy_port=None*, *proxy_user=None*, *proxy_pass=None*, *debug=0*, *https_connection_factory=None*, *path='/'*, *provider='aws'*, *security_token=None*, *suppress_consec_slashes=True*)

Parameters

- **`host`** (*`str`*) – The host to make the connection to
- **`aws_access_key_id`** (*`str`*) – Your AWS Access Key ID (provided by Amazon). If none is specified, the value in your `AWS_ACCESS_KEY_ID` environmental variable is used.
- **`aws_secret_access_key`** (*`str`*) – Your AWS Secret Access Key (provided by Amazon). If none is specified, the value in your `AWS_SECRET_ACCESS_KEY` environmental variable is used.
- **`is_secure`** (*`boolean`*) – Whether the connection is over SSL
- **`https_connection_factory`** (*`list or tuple`*) – A pair of an HTTP connection factory and the exceptions to catch. The factory should have a similar interface to L{httplib.HTTPSConnection}.
- **`proxy`** (*`str`*) – Address/hostname for a proxy server
- **`proxy_port`** (*`int`*) – The port to use when connecting over a proxy

- **proxy_user** (*str*) – The username to connect with on the proxy
- **proxy_pass** (*str*) – The password to use when connection over a proxy.
- **port** (*int*) – The port to use to connect
- **suppress_consec_slashes** (*bool*) – If provided, controls whether consecutive slashes will be suppressed in key paths.

access_key

aws_access_key_id

aws_secret_access_key

build_base_http_request(*method*, *path*, *auth_path*, *params=None*, *headers=None*, *data=''*, *host=None*)

close()
(Optional) Close any open HTTP connections. This is non-destructive, and making a new request will open a connection again.

connection

get_http_connection(*host*, *is_secure*)

get_path(*path='/'*)

get_proxy_auth_header()

gs_access_key_id

gs_secret_access_key

handle_proxy(*proxy*, *proxy_port*, *proxy_user*, *proxy_pass*)

make_request(*method*, *path*, *headers=None*, *data=''*, *host=None*, *auth_path=None*, *sender=None*, *override_num_retries=None*)
Makes a request to the server, with stock multiple-retry logic.

new_http_connection(*host*, *is_secure*)

prefix_proxy_to_path(*path*, *host=None*)

proxy_ssl()

put_http_connection(*host*, *is_secure*, *connection*)

secret_key

server_name(*port=None*)

class boto.connection.**AWSQueryConnection**(*aws_access_key_id=None*, *aws_secret_access_key=None*, *is_secure=True*, *port=None*, *proxy=None*, *proxy_port=None*, *proxy_user=None*, *proxy_pass=None*, *host=None*, *debug=0*, *https_connection_factory=None*, *path='/'*, *security_token=None*)

APIVersion = ''

ResponseError
alias of BotoServerError

build_list_params(*params*, *items*, *label*)

get_list(*action*, *params*, *markers*, *path='/'*, *parent=None*, *verb='GET'*)

get_object (*action*, *params*, *cls*, *path='/'*, *parent=None*, *verb='GET'*)

get_status (*action*, *params*, *path='/'*, *parent=None*, *verb='GET'*)

get_utf8_value (*value*)

make_request (*action*, *params=None*, *path='/'*, *verb='GET'*)

class boto.connection.**ConnectionPool**

A connection pool that expires connections after a fixed period of time. This saves time spent waiting for a connection that AWS has timed out on the other end.

This class is thread-safe.

CLEAN_INTERVAL = 5.0

STALE_DURATION = 60.0

clean ()

Clean up the stale connections in all of the pools, and then get rid of empty pools. Pools clean themselves every time a connection is fetched; this cleaning takes care of pools that aren't being used any more, so nothing is being gotten from them.

get_http_connection (*host*, *is_secure*)

Gets a connection from the pool for the named host. Returns None if there is no connection that can be reused. It's the caller's responsibility to call close() on the connection when it's no longer needed.

put_http_connection (*host*, *is_secure*, *conn*)

Adds a connection to the pool of connections that can be reused for the named host.

size ()

Returns the number of connections in the pool.

class boto.connection.**HTTPRequest** (*method*, *protocol*, *host*, *port*, *path*, *auth_path*, *params*, *headers*, *body*)

Represents an HTTP request.

Parameters

- **method** (*string*) – The HTTP method name, 'GET', 'POST', 'PUT' etc.
- **protocol** (*string*) – The http protocol used, 'http' or 'https'.
- **host** (*string*) – Host to which the request is addressed. eg. abc.com
- **port** (*int*) – port on which the request is being sent. Zero means unset, in which case default port will be chosen.
- **path** (*string*) – URL path that is being accessed.
- **path** – The part of the URL path used when creating the authentication string.
- **params** (dict) – HTTP url query parameters, with key as name of the param, and value as value of param.
- **headers** (dict) – HTTP headers, with key as name of the header and value as value of header.
- **body** (*string*) – Body of the HTTP request. If not present, will be None or empty string ('').

authorize (*connection*, ***kwargs*)

class boto.connection.**HostConnectionPool**

A pool of connections for one remote (host,is_secure).

When connections are added to the pool, they are put into a pending queue. The _mexe method returns connections to the pool before the response body has been read, so they connections aren't ready to send another request yet. They stay in the pending queue until they are ready for another request, at which point they are returned to the pool of ready connections.

The pool of ready connections is an ordered list of (connection,time) pairs, where the time is the time the connection was returned from _mexe. After a certain period of time, connections are considered stale, and discarded rather than being reused. This saves having to wait for the connection to time out if AWS has decided to close it on the other end because of inactivity.

Thread Safety:

This class is used only fram ConnectionPool while it's mutex is held.

clean()
Get rid of stale connections.

get()
Returns the next connection in this pool that is ready to be reused. Returns None of there aren't any.

put(*conn*)
Adds a connection to the pool, along with the time it was added.

size()
Returns the number of connections in the pool for this host. Some of the connections may still be in use, and may not be ready to be returned by get().

boto.exception

Exception classes - Subclassing allows you to check for specific errors

exception boto.exception.**AWSConnectionError**(*reason*, **args*)
General error connecting to Amazon Web Services.

exception boto.exception.**BotoClientError**(*reason*, **args*)
General Boto Client error (error accessing AWS)

exception boto.exception.**BotoServerError**(*status*, *reason*, *body=None*, **args*)

endElement(*name*, *value*, *connection*)

startElement(*name*, *attrs*, *connection*)

class boto.exception.**ConsoleOutput**(*parent=None*)

endElement(*name*, *value*, *connection*)

startElement(*name*, *attrs*, *connection*)

exception boto.exception.**DynamoDBResponseError**(*status*, *reason*, *data*)

exception boto.exception.**EC2ResponseError**(*status*, *reason*, *body=None*)
Error in response from EC2.

endElement(*name*, *value*, *connection*)

startElement(*name*, *attrs*, *connection*)

exception boto.exception.**EmrResponseError**(*status*, *reason*, *body=None*, **args*)
Error in response from EMR

exception boto.exception.**FPSResponseError**(*status*, *reason*, *body=None*, **args*)

exception `boto.exception.`**`GSCopyError`**(*status*, *reason*, *body=None*, **args*)
Error copying a key on GS.

exception `boto.exception.`**`GSCreateError`**(*status*, *reason*, *body=None*)
Error creating a bucket or key on GS.

exception `boto.exception.`**`GSDataError`**(*reason*, **args*)
Error receiving data from GS.

exception `boto.exception.`**`GSPermissionsError`**(*reason*, **args*)
Permissions error when accessing a bucket or key on GS.

exception `boto.exception.`**`GSResponseError`**(*status*, *reason*, *body=None*)
Error in response from GS.

exception `boto.exception.`**`InvalidAclError`**(*message*)
Exception raised when ACL XML is invalid.

exception `boto.exception.`**`InvalidUriError`**(*message*)
Exception raised when URI is invalid.

exception `boto.exception.`**`NoAuthHandlerFound`**
Is raised when no auth handlers were found ready to authenticate.

exception `boto.exception.`**`ResumableDownloadException`**(*message*, *disposition*)
Exception raised for various resumable download problems.

self.disposition is of type ResumableTransferDisposition.

class `boto.exception.`**`ResumableTransferDisposition`**

`ABORT` = 'ABORT'

`ABORT_CUR_PROCESS` = 'ABORT_CUR_PROCESS'

`START_OVER` = 'START_OVER'

`WAIT_BEFORE_RETRY` = 'WAIT_BEFORE_RETRY'

exception `boto.exception.`**`ResumableUploadException`**(*message*, *disposition*)
Exception raised for various resumable upload problems.

self.disposition is of type ResumableTransferDisposition.

exception `boto.exception.`**`S3CopyError`**(*status*, *reason*, *body=None*, **args*)
Error copying a key on S3.

exception `boto.exception.`**`S3CreateError`**(*status*, *reason*, *body=None*)
Error creating a bucket or key on S3.

exception `boto.exception.`**`S3DataError`**(*reason*, **args*)
Error receiving data from S3.

exception `boto.exception.`**`S3PermissionsError`**(*reason*, **args*)
Permissions error when accessing a bucket or key on S3.

exception `boto.exception.`**`S3ResponseError`**(*status*, *reason*, *body=None*)
Error in response from S3.

exception `boto.exception.`**`SDBPersistenceError`**

exception `boto.exception.`**`SDBResponseError`**(*status*, *reason*, *body=None*, **args*)
Error in responses from SDB.

exception boto.exception.**SQSDecodeError**(*reason*, *message*)
    Error when decoding an SQS message.

exception boto.exception.**SQSError**(*status*, *reason*, *body=None*)
    General Error on Simple Queue Service.

    endElement(*name*, *value*, *connection*)

    startElement(*name*, *attrs*, *connection*)

exception boto.exception.**StorageCopyError**(*status*, *reason*, *body=None*, **args*)
    Error copying a key on a storage service.

exception boto.exception.**StorageCreateError**(*status*, *reason*, *body=None*)
    Error creating a bucket or key on a storage service.

    endElement(*name*, *value*, *connection*)

exception boto.exception.**StorageDataError**(*reason*, **args*)
    Error receiving data from a storage service.

exception boto.exception.**StoragePermissionsError**(*reason*, **args*)
    Permissions error when accessing a bucket or key on a storage service.

exception boto.exception.**StorageResponseError**(*status*, *reason*, *body=None*)
    Error in response from a storage service.

    endElement(*name*, *value*, *connection*)

    startElement(*name*, *attrs*, *connection*)

exception boto.exception.**TooManyAuthHandlerReadyToAuthenticate**
    Is raised when there are more than one auth handler ready.

    In normal situation there should only be one auth handler that is ready to authenticate. In case where more than one auth handler is ready to authenticate, we raise this exception, to prevent unpredictable behavior when multiple auth handlers can handle a particular case and the one chosen depends on the order they were checked.

boto.handler

class boto.handler.**XmlHandler**(*root_node*, *connection*)

    characters(*content*)

    endElement(*name*)

    startElement(*name*, *attrs*)

boto.resultset

class boto.resultset.**BooleanResult**(*marker_elem=None*)

    endElement(*name*, *value*, *connection*)

    startElement(*name*, *attrs*, *connection*)

    to_boolean(*value*, *true_value='true'*)

class `boto.resultset.`**`ResultSet`**(*marker_elem=None*)

The ResultSet is used to pass results back from the Amazon services to the client. It is light wrapper around Python's `list` class, with some additional methods for parsing XML results from AWS. Because I don't really want any dependencies on external libraries, I'm using the standard SAX parser that comes with Python. The good news is that it's quite fast and efficient but it makes some things rather difficult.

You can pass in, as the marker_elem parameter, a list of tuples. Each tuple contains a string as the first element which represents the XML element that the resultset needs to be on the lookout for and a Python class as the second element of the tuple. Each time the specified element is found in the XML, a new instance of the class will be created and popped onto the stack.

Variables ***`next_token`*** (*`str`*) – A hash used to assist in paging through very long result sets. In most cases, passing this value to certain methods will give you another 'page' of results.

`endElement`(*name*, *value*, *connection*)

`startElement`(*name*, *attrs*, *connection*)

`to_boolean`(*value*, *true_value='true'*)

boto.utils

Some handy utility functions used by several classes.

class `boto.utils.`**`AuthSMTPHandler`**(*mailhost*, *username*, *password*, *fromaddr*, *toaddrs*, *subject*)

This class extends the SMTPHandler in the standard Python logging module to accept a username and password on the constructor and to then use those credentials to authenticate with the SMTP server. To use this, you could add something like this in your boto config file:

[handler_hand07] class=boto.utils.AuthSMTPHandler level=WARN formatter=form07 args=('localhost', 'username', 'password', 'from@abc', ['user1@abc', 'user2@xyz'], 'Logger Subject')

Initialize the handler.

We have extended the constructor to accept a username/password for SMTP authentication.

`emit`(*record*)

Emit a record.

Format the record and send it to the specified addressees. It would be really nice if I could add authorization to this class without having to resort to cut and paste inheritance but, no.

class `boto.utils.`**`LRUCache`**(*capacity*)

A dictionary-like object that stores only a certain number of items, and discards its least recently used item when full.

```
>>> cache = LRUCache(3)
>>> cache['A'] = 0
>>> cache['B'] = 1
>>> cache['C'] = 2
>>> len(cache)
3
```

```
>>> cache['A']
0
```

Adding new items to the cache does not increase its size. Instead, the least recently used item is dropped:

```
>>> cache['D'] = 3
>>> len(cache)
3
>>> 'B' in cache
False
```

Iterating over the cache returns the keys, starting with the most recently used:

```
>>> for key in cache:
...     print key
D
A
C
```

This code is based on the LRUCache class from Genshi which is based on Mighty's LRUCache from `myghtyutils.util`, written by Mike Bayer and released under the MIT license (Genshi uses the BSD License). See:

> http://svn.myghty.org/myghtyutils/trunk/lib/myghtyutils/util.py

class `boto.utils.`**`Password`**(*str=None*, *hashfunc=None*)
Password object that stores itself as hashed. Hash defaults to SHA512 if available, MD5 otherwise.

Load the string from an initial value, this should be the raw hashed password.

`hashfunc`()
Returns a sha512 hash object; optionally initialized with a string

`set`(*value*)

class `boto.utils.`**`ShellCommand`**(*command*, *wait=True*, *fail_fast=False*, *cwd=None*)

`getOutput`()

`getStatus`()

`output`
The STDIN and STDERR output of the command

`run`(*cwd=None*)

`setReadOnly`(*value*)

`status`
The exit code for the command

`boto.utils.`**`canonical_string`**(*method*, *path*, *headers*, *expires=None*, *provider=None*)

`boto.utils.`**`compute_md5`**(*fp*, *buf_size=8192*, *size=None*)
Compute MD5 hash on passed file and return results in a tuple of values.

Parameters

- **fp** (`file`) – File pointer to the file to MD5 hash. The file pointer will be reset to its current location before the method returns.
- **buf_size** (*integer*) – Number of bytes per read request.
- **size** (*int*) – (optional) The Maximum number of bytes to read from the file pointer (fp). This is useful when uploading a file in multiple parts where the file is being split inplace into different parts. Less bytes may be available.

Return type tuple

Returns A tuple containing the hex digest version of the MD5 hash as the first element, the base64 encoded version of the plain digest as the second element and the data size as the third element.

`boto.utils.`**`fetch_file`**(*uri*, *file=None*, *username=None*, *password=None*)
Fetch a file based on the URI provided. If you do not pass in a file pointer a tempfile.NamedTemporaryFile, or None if the file could not be retrieved is returned. The URI can be either an HTTP url, or "s3://bucket_name/key_name"

`boto.utils.`**`find_class`**(*module_name*, *class_name=None*)

`boto.utils.`**`get_aws_metadata`**(*headers*, *provider=None*)

`boto.utils.`**`get_instance_metadata`**(*version='latest'*, *url='http://169.254.169.254'*)
Returns the instance metadata as a nested Python dictionary. Simple values (e.g. local_hostname, hostname, etc.) will be stored as string values. Values such as ancestor-ami-ids will be stored in the dict as a list of string values. More complex fields such as public-keys and will be stored as nested dicts.

`boto.utils.`**`get_instance_userdata`**(*version='latest'*, *sep=None*, *url='http://169.254.169.254'*)

`boto.utils.`**`get_ts`**(*ts=None*)

`boto.utils.`**`get_utf8_value`**(*value*)

`boto.utils.`**`guess_mime_type`**(*content*, *deftype*)
Description: Guess the mime type of a block of text :param content: content we're finding the type of :type str:

Parameters **`deftype`** – Default mime type

Return type <type>:

Returns <description>

`boto.utils.`**`merge_meta`**(*headers*, *metadata*, *provider=None*)

`boto.utils.`**`mklist`**(*value*)

`boto.utils.`**`notify`**(*subject*, *body=None*, *html_body=None*, *to_string=None*, *attachments=None*, *append_instance_id=True*)

`boto.utils.`**`parse_ts`**(*ts*)

`boto.utils.`**`pythonize_name`**(*name*, *sep='_'*)

`boto.utils.`**`retry_url`**(*url*, *retry_on_404=True*, *num_retries=10*)

`boto.utils.`**`unquote_v`**(*nv*)

`boto.utils.`**`update_dme`**(*username*, *password*, *dme_id*, *ip_address*)
Update your Dynamic DNS record with DNSMadeEasy.com

`boto.utils.`**`write_mime_multipart`**(*content*, *compress=False*, *deftype='text/plain'*, *delimiter=':'*)
Description: :param content: A list of tuples of name-content pairs. This is used instead of a dict to ensure that scripts run in order :type list of tuples:

Parameters

- **`compress`** – Use gzip to compress the scripts, defaults to no compression
- **`deftype`** – The type that should be assumed if nothing else can be figured out
- **`delimiter`** – mime delimiter

Returns Final mime multipart

Return type str:

contrib

boto.contrib

boto.contrib.m2helpers

Note: This module requires installation of M2Crypto in your Python path.

boto.contrib.ymlmessage

This module was contributed by Chris Moyer. It provides a subclass of the SQS Message class that supports YAML as the body of the message.

This module requires the yaml module.

class `boto.contrib.ymlmessage.`**`YAMLMessage`**(*queue=None*, *body=''*, *xml_attrs=None*)

The YAMLMessage class provides a YAML compatible message. Encoding and decoding are handled automaticaly.

Access this message data like such:

m.data = [1, 2, 3] m.data[0] # Returns 1

This depends on the PyYAML package

`get_body`()

`set_body`(*body*)

ECS

boto.ecs

class `boto.ecs.`**`ECSConnection`**(*aws_access_key_id=None*, *aws_secret_access_key=None*, *is_secure=True*, *port=None*, *proxy=None*, *proxy_port=None*, *proxy_user=None*, *proxy_pass=None*, *host='ecs.amazonaws.com'*, *debug=0*, *https_connection_factory=None*, *path='/'*)

ECommerce Connection

For more information on how to use this module see:

http://blog.coredumped.org/2010/09/search-for-books-on-amazon-using-boto.html

`APIVersion` **= '2010-11-01'**

`get_response`(*action*, *params*, *page=0*, *itemSet=None*)

Utility method to handle calls to ECS and parsing of responses.

`item_search`(*search_index*, ***params*)

Returns items that satisfy the search criteria, including one or more search indices.

For a full list of search terms, :see: http://docs.amazonwebservices.com/AWSECommerceService/2010-09-01/DG/index.html?ItemSearch.html

boto.ecs.item

class `boto.ecs.item.`**`Item`**(*connection=None*)

A single Item

Initialize this Item

class `boto.ecs.item.`**`ItemSet`**(*connection*, *action*, *params*, *page=0*)

A special ResponseGroup that has built-in paging, and only creates new Items on the "Item" tag

`endElement`(*name*, *value*, *connection*)

`next`()

Special paging functionality

`startElement`(*name*, *attrs*, *connection*)

`to_xml`()

Override to first fetch everything

class `boto.ecs.item.`**`ResponseGroup`**(*connection=None*, *nodename=None*)

A Generic "Response Group", which can be anything from the entire list of Items to specific response elements within an item

Initialize this Item

`endElement`(*name*, *value*, *connection*)

`get`(*name*)

`set`(*name*, *value*)

`startElement`(*name*, *attrs*, *connection*)

`to_xml`()

file

boto.file.bucket

class `boto.file.bucket.`**`Bucket`**(*name*, *contained_key*)

Instantiate an anonymous file-based Bucket around a single key.

`delete_key`(*key_name*, *headers=None*, *version_id=None*, *mfa_token=None*)

Deletes a key from the bucket.

Parameters

- **`key_name`** (`string`) – The key name to delete
- **`version_id`** (`string`) – Unused in this subclass.
- **`mfa_token`** (`tuple or list of strings`) – Unused in this subclass.

`get_all_keys`(*headers=None*, ***params*)

This method returns the single key around which this anonymous Bucket was instantiated.

Return type *SimpleResultSet*

Returns The result from file system listing the keys requested

`get_key`(*key_name*, *headers=None*, *version_id=None*, *key_type=0*)

Check to see if a particular key exists within the bucket. Returns: An instance of a Key object or None

Parameters

- **key_name** (*string*) – The name of the key to retrieve
- **version_id** (*string*) – Unused in this subclass.
- **stream_type** (*integer*) – Type of the Key - Regular File or input/output Stream

Return type *boto.file.key.Key*

Returns A Key object from this bucket.

new_key (*key_name=None*, *key_type=0*)
Creates a new key

Parameters **key_name** (*string*) – The name of the key to create

Return type *boto.file.key.Key*

Returns An instance of the newly created key object

boto.file.simpleresultset

class boto.file.simpleresultset.**SimpleResultSet** (*input_list*)
ResultSet facade built from a simple list, rather than via XML parsing.

boto.file.connection

class boto.file.connection.**FileConnection** (*file_storage_uri*)

get_bucket (*bucket_name*, *validate=True*, *headers=None*)

boto.file.key

class boto.file.key.**Key** (*bucket*, *name*, *fp=None*, *key_type=0*)

KEY_REGULAR_FILE = 0

KEY_STREAM = 3

KEY_STREAM_READABLE = 1

KEY_STREAM_WRITABLE = 2

close ()
Closes fp associated with underlying file. Caller should call this method when done with this class, to avoid using up OS resources (e.g., when iterating over a large number of files).

get_contents_as_string (*headers=None*, *cb=None*, *num_cb=10*, *torrent=False*)
Retrieve file data from the Key, and return contents as a string.

Parameters

- **headers** (dict) – ignored in this subclass.
- **cb** (*int*) – ignored in this subclass.
- **num_cb** – ignored in this subclass.
- **num_cb** – ignored in this subclass.

- **torrent** (*bool*) – ignored in this subclass.

Return type string

Returns The contents of the file as a string

get_file(*fp*, *headers=None*, *cb=None*, *num_cb=10*, *torrent=False*)
Retrieves a file from a Key

Parameters

- **fp** (file) – File pointer to put the data into
- **cb** (*int*) – ignored in this subclass.
- **num_cb** – ignored in this subclass.

Param ignored in this subclass.

is_stream()

set_contents_from_file(*fp*, *headers=None*, *replace=True*, *cb=None*, *num_cb=10*, *policy=None*, *md5=None*)
Store an object in a file using the name of the Key object as the key in file URI and the contents of the file pointed to by 'fp' as the contents.

Parameters

- **fp** (file) – the file whose contents to upload
- **headers** (dict) – ignored in this subclass.
- **replace** (*bool*) – If this parameter is False, the method will first check to see if an object exists in the bucket with the same key. If it does, it won't overwrite it. The default value is True which will overwrite the object.
- **cb** (*int*) – ignored in this subclass.
- **num_cb** – ignored in this subclass.
- **policy** (boto.s3.acl.CannedACLStrings) – ignored in this subclass.
- **md5** (*A tuple containing the hexdigest version of the MD5 checksum of the file as the first element and the Base64-encoded version of the plain checksum as the second element. This is the same format returned by the compute_md5 method.*) – ignored in this subclass.

GS

boto.gs.acl

class boto.gs.acl.**ACL**(*parent=None*)

acl

add_email_grant(*permission*, *email_address*)

add_group_email_grant(*permission*, *email_address*)

add_group_grant(*permission*, *group_id*)

add_user_grant(*permission*, *user_id*)

endElement (*name*, *value*, *connection*)

startElement (*name*, *attrs*, *connection*)

to_xml ()

class boto.gs.acl.**Entries** (*parent=None*)

endElement (*name*, *value*, *connection*)

startElement (*name*, *attrs*, *connection*)

to_xml ()

class boto.gs.acl.**Entry** (*scope=None*, *type=None*, *id=None*, *name=None*, *email_address=None*, *domain=None*, *permission=None*)

endElement (*name*, *value*, *connection*)

startElement (*name*, *attrs*, *connection*)

to_xml ()

class boto.gs.acl.**Scope** (*parent*, *type=None*, *id=None*, *name=None*, *email_address=None*, *domain=None*)

ALLOWED_SCOPE_TYPE_SUB_ELEMS = {'GroupByDomain': ['Domain'], 'UserByEmail': ['EmailAddress', 'Name'], 'U

endElement (*name*, *value*, *connection*)

startElement (*name*, *attrs*, *connection*)

to_xml ()

boto.gs.bucket

class boto.gs.bucket.**Bucket** (*connection=None*, *name=None*, *key_class=<class 'boto.gs.key.Key'>*)

add_email_grant (*permission*, *email_address*, *recursive=False*, *headers=None*)

Convenience method that provides a quick way to add an email grant to a bucket. This method retrieves the current ACL, creates a new grant based on the parameters passed in, adds that grant to the ACL and then PUT's the new ACL back to GS.

Parameters

- **permission** (*string*) – The permission being granted. Should be one of: (READ, WRITE, FULL_CONTROL).
- **email_address** (*string*) – The email address associated with the GS account your are granting the permission to.
- **recursive** (*boolean*) – A boolean value to controls whether the call will apply the grant to all keys within the bucket or not. The default value is False. By passing a True value, the call will iterate through all keys in the bucket and apply the same grant to each key. CAUTION: If you have a lot of keys, this could take a long time!

add_group_email_grant (*permission*, *email_address*, *recursive=False*, *headers=None*)

Convenience method that provides a quick way to add an email group grant to a bucket. This method retrieves the current ACL, creates a new grant based on the parameters passed in, adds that grant to the ACL and then PUT's the new ACL back to GS.

Parameters

- **permission** (*string*) – The permission being granted. Should be one of: READ|WRITE|FULL_CONTROL See http://code.google.com/apis/storage/docs/developer-guide.html#authorization for more details on permissions.
- **email_address** (*string*) – The email address associated with the Google Group to which you are granting the permission.
- **recursive** (*bool*) – A boolean value to controls whether the call will apply the grant to all keys within the bucket or not. The default value is False. By passing a True value, the call will iterate through all keys in the bucket and apply the same grant to each key. CAUTION: If you have a lot of keys, this could take a long time!

add_user_grant(*permission*, *user_id*, *recursive=False*, *headers=None*)
Convenience method that provides a quick way to add a canonical user grant to a bucket. This method retrieves the current ACL, creates a new grant based on the parameters passed in, adds that grant to the ACL and then PUTs the new ACL back to GS.

Parameters

- **permission** (*string*) – The permission being granted. Should be one of: (READ|WRITE|FULL_CONTROL)
- **user_id** (*string*) – The canonical user id associated with the GS account you are granting the permission to.
- **recursive** (*bool*) – A boolean value to controls whether the call will apply the grant to all keys within the bucket or not. The default value is False. By passing a True value, the call will iterate through all keys in the bucket and apply the same grant to each key. CAUTION: If you have a lot of keys, this could take a long time!

disable_logging(*headers=None*)

enable_logging(*target_bucket*, *target_prefix=None*, *headers=None*)

get_acl(*key_name=''*, *headers=None*, *version_id=None*)
returns a bucket's acl. We include a version_id argument to support a polymorphic interface for callers, however, version_id is not relevant for Google Cloud Storage buckets and is therefore ignored here.

get_acl_helper(*key_name*, *headers*, *query_args*)
provides common functionality for get_acl() and get_def_acl()

get_def_acl(*key_name=''*, *headers=None*)
returns a bucket's default object acl

list_grants(*headers=None*)

set_acl(*acl_or_str*, *key_name=''*, *headers=None*, *version_id=None*)
sets or changes a bucket's acl. We include a version_id argument to support a polymorphic interface for callers, however, version_id is not relevant for Google Cloud Storage buckets and is therefore ignored here.

set_canned_acl(*acl_str*, *key_name=''*, *headers=None*, *version_id=None*)
sets or changes a bucket's acl to a predefined (canned) value. We include a version_id argument to support a polymorphic interface for callers, however, version_id is not relevant for Google Cloud Storage buckets and is therefore ignored here.

set_canned_acl_helper(*acl_str*, *key_name*, *headers*, *query_args*)
provides common functionality for set_canned_acl() and set_def_canned_acl()

set_def_acl(*acl_or_str*, *key_name=''*, *headers=None*)
sets or changes a bucket's default object acl

set_def_canned_acl(*acl_str*, *key_name=''*, *headers=None*)
sets or changes a bucket's default object acl to a predefined (canned) value

set_def_xml_acl(*acl_str*, *key_name=''*, *headers=None*)
sets or changes a bucket's default object

boto.gs.connection

class boto.gs.connection.**GSConnection**(*gs_access_key_id=None*, *gs_secret_access_key=None*, *is_secure=True*, *port=None*, *proxy=None*, *proxy_port=None*, *proxy_user=None*, *proxy_pass=None*, *host='commondatastorage.googleapis.com'*, *debug=0*, *https_connection_factory=None*, *calling_format=<boto.s3.connection.SubdomainCallingFormat object>*, *path='/'*, *suppress_consec_slashes=True*)

DefaultHost = 'commondatastorage.googleapis.com'

QueryString = 'Signature=%s&Expires=%d&AWSAccessKeyId=%s'

create_bucket(*bucket_name*, *headers=None*, *location=''*, *policy=None*)
Creates a new bucket. By default it's located in the USA. You can pass Location.EU to create an European bucket. You can also pass a LocationConstraint, which (in addition to locating the bucket in the specified location) informs Google that Google services must not copy data out of that location.

Parameters

- **bucket_name** (*string*) – The name of the new bucket
- **headers** (dict) – Additional headers to pass along with the request to AWS.
- **location** (*boto.gs.connection.Location*) – The location of the new bucket
- **policy** (boto.s3.acl.CannedACLStrings) – A canned ACL policy that will be applied to the new key in S3.

class boto.gs.connection.**Location**

DEFAULT = ''

EU = 'EU'

boto.gs.key

class boto.gs.key.**Key**(*bucket=None*, *name=None*)

add_email_grant(*permission*, *email_address*)
Convenience method that provides a quick way to add an email grant to a key. This method retrieves the current ACL, creates a new grant based on the parameters passed in, adds that grant to the ACL and then PUT's the new ACL back to GS.

Parameters

- **permission** (*string*) – The permission being granted. Should be one of: READ|FULL_CONTROL See http://code.google.com/apis/storage/docs/developer-guide.html#authorization for more details on permissions.
- **email_address** (*string*) – The email address associated with the Google account to which you are granting the permission.

add_group_email_grant (*permission*, *email_address*, *headers=None*)
Convenience method that provides a quick way to add an email group grant to a key. This method retrieves the current ACL, creates a new grant based on the parameters passed in, adds that grant to the ACL and then PUT's the new ACL back to GS.

Parameters

- **permission** (*string*) – The permission being granted. Should be one of: READ|FULL_CONTROL See http://code.google.com/apis/storage/docs/developer-guide.html#authorization for more details on permissions.
- **email_address** (*string*) – The email address associated with the Google Group to which you are granting the permission.

add_group_grant (*permission*, *group_id*)
Convenience method that provides a quick way to add a canonical group grant to a key. This method retrieves the current ACL, creates a new grant based on the parameters passed in, adds that grant to the ACL and then PUT's the new ACL back to GS.

Parameters

- **permission** (*string*) – The permission being granted. Should be one of: READ|FULL_CONTROL See http://code.google.com/apis/storage/docs/developer-guide.html#authorization for more details on permissions.
- **group_id** (*string*) – The canonical group id associated with the Google Groups account you are granting the permission to.

add_user_grant (*permission*, *user_id*)
Convenience method that provides a quick way to add a canonical user grant to a key. This method retrieves the current ACL, creates a new grant based on the parameters passed in, adds that grant to the ACL and then PUT's the new ACL back to GS.

Parameters

- **permission** (*string*) – The permission being granted. Should be one of: READ|FULL_CONTROL See http://code.google.com/apis/storage/docs/developer-guide.html#authorization for more details on permissions.
- **user_id** (*string*) – The canonical user id associated with the GS account to which you are granting the permission.

set_contents_from_file (*fp*, *headers=None*, *replace=True*, *cb=None*, *num_cb=10*, *policy=None*, *md5=None*, *res_upload_handler=None*, *size=None*)
Store an object in GS using the name of the Key object as the key in GS and the contents of the file pointed to by 'fp' as the contents.

Parameters

- **fp** (file) – the file whose contents are to be uploaded
- **headers** (dict) – additional HTTP headers to be sent with the PUT request.
- **replace** (*bool*) – If this parameter is False, the method will first check to see if an object exists in the bucket with the same key. If it does, it won't overwrite it. The default value is True which will overwrite the object.
- **cb** (*function*) – a callback function that will be called to report progress on the upload. The callback should accept two integer parameters, the first representing the number of bytes that have been successfully transmitted to GS and the second representing the total number of bytes that need to be transmitted.

- **num_cb** (*int*) – (optional) If a callback is specified with the cb parameter, this parameter determines the granularity of the callback by defining the maximum number of times the callback will be called during the file transfer.
- **policy** (`boto.gs.acl.CannedACLStrings`) – A canned ACL policy that will be applied to the new key in GS.
- **md5** (*A tuple containing the hexdigest version of the MD5 checksum of the file as the first element and the Base64-encoded version of the plain checksum as the second element. This is the same format returned by the compute_md5 method.*) – If you need to compute the MD5 for any reason prior to upload, it's silly to have to do it twice so this param, if present, will be used as the MD5 values of the file. Otherwise, the checksum will be computed.
- **res_upload_handler** (`ResumableUploadHandler`) – If provided, this handler will perform the upload.
- **size** (*int*) – (optional) The Maximum number of bytes to read from the file pointer (fp). This is useful when uploading a file in multiple parts where you are splitting the file up into different ranges to be uploaded. If not specified, the default behaviour is to read all bytes from the file pointer. Less bytes may be available. Notes:
  1. The "size" parameter currently cannot be used when a resumable upload handler is given but is still useful for uploading part of a file as implemented by the parent class.
  2. At present Google Cloud Storage does not support multipart uploads.

TODO: At some point we should refactor the Bucket and Key classes, to move functionality common to all providers into a parent class, and provider-specific functionality into subclasses (rather than just overriding/sharing code the way it currently works).

set_contents_from_filename (*filename*, *headers=None*, *replace=True*, *cb=None*, *num_cb=10*, *policy=None*, *md5=None*, *reduced_redundancy=None*, *res_upload_handler=None*)

Store an object in GS using the name of the Key object as the key in GS and the contents of the file named by 'filename'. See set_contents_from_file method for details about the parameters.

Parameters

- **filename** (*string*) – The name of the file that you want to put onto GS
- **headers** (`dict`) – Additional headers to pass along with the request to GS.
- **replace** (*bool*) – If True, replaces the contents of the file if it already exists.
- **cb** (*int*) – (optional) a callback function that will be called to report progress on the download. The callback should accept two integer parameters, the first representing the number of bytes that have been successfully transmitted from GS and the second representing the total number of bytes that need to be transmitted.
- **num_cb** – (optional) If a callback is specified with the cb parameter this parameter determines the granularity of the callback by defining the maximum number of times the callback will be called during the file transfer.
- **policy** (`boto.gs.acl.CannedACLStrings`) – A canned ACL policy that will be applied to the new key in GS.
- **md5** (*A tuple containing the hexdigest version of the MD5 checksum of the file as the first element and the Base64-encoded version of the plain checksum as the second element. This is the same format returned by the*

compute_md5 method.) – If you need to compute the MD5 for any reason prior to upload, it's silly to have to do it twice so this param, if present, will be used as the MD5 values of the file. Otherwise, the checksum will be computed.

- **res_upload_handler** (ResumableUploadHandler) – If provided, this handler will perform the upload.

set_contents_from_string(*s*, *headers=None*, *replace=True*, *cb=None*, *num_cb=10*, *policy=None*, *md5=None*)

Store an object in S3 using the name of the Key object as the key in S3 and the string 's' as the contents. See set_contents_from_file method for details about the parameters.

Parameters

- **headers** (dict) – Additional headers to pass along with the request to AWS.
- **replace** (*bool*) – If True, replaces the contents of the file if it already exists.
- **cb** (*int*) – a callback function that will be called to report progress on the upload. The callback should accept two integer parameters, the first representing the number of bytes that have been successfully transmitted to S3 and the second representing the size of the to be transmitted object.
- **num_cb** – (optional) If a callback is specified with the cb parameter this parameter determines the granularity of the callback by defining the maximum number of times the callback will be called during the file transfer.
- **policy** (boto.s3.acl.CannedACLStrings) – A canned ACL policy that will be applied to the new key in S3.
- **md5** (*A tuple containing the hexdigest version of the MD5 checksum of the file as the first element and the Base64-encoded version of the plain checksum as the second element. This is the same format returned by the compute_md5 method.*) – If you need to compute the MD5 for any reason prior to upload, it's silly to have to do it twice so this param, if present, will be used as the MD5 values of the file. Otherwise, the checksum will be computed.

boto.gs.user

class boto.gs.user.**User**(*parent=None*, *id=''*, *name=''*)

endElement(*name*, *value*, *connection*)

startElement(*name*, *attrs*, *connection*)

to_xml(*element_name='Owner'*)

boto.gs.resumable_upload_handler

class boto.gs.resumable_upload_handler.**ResumableUploadHandler**(*tracker_file_name=None*, *num_retries=None*)

Constructor. Instantiate once for each uploaded file.

Parameters

- **tracker_file_name** (*string*) – optional file name to save tracker URI. If supplied and the current process fails the upload, it can be retried in a new process. If called with an existing file containing a valid tracker URI, we'll resume the upload from this URI; else we'll start a new resumable upload (and write the URI to this tracker file).

- **num_retries** (*int*) – the number of times we'll re-try a resumable upload making no progress. (Count resets every time we get progress, so upload can span many more than this number of retries.)

BUFFER_SIZE = 8192

RETRYABLE_EXCEPTIONS = (<class 'httplib.HTTPException'>, <type 'exceptions.IOError'>, <class 'socket.error'>, <cl

SERVER_HAS_NOTHING = (0, -1)

get_tracker_uri()
Returns upload tracker URI, or None if the upload has not yet started.

send_file(*key*, *fp*, *headers*, *cb=None*, *num_cb=10*)
Upload a file to a key into a bucket on GS, using GS resumable upload protocol.

Parameters

- **key** (`boto.s3.key.Key` or subclass) – The Key object to which data is to be uploaded
- **fp** (*file-like object*) – The file pointer to upload
- **headers** (*dict*) – The headers to pass along with the PUT request
- **cb** (*function*) – a callback function that will be called to report progress on the upload. The callback should accept two integer parameters, the first representing the number of bytes that have been successfully transmitted to GS, and the second representing the total number of bytes that need to be transmitted.
- **num_cb** (*int*) – (optional) If a callback is specified with the cb parameter, this parameter determines the granularity of the callback by defining the maximum number of times the callback will be called during the file transfer. Providing a negative integer will cause your callback to be called with each buffer read.

Raises ResumableUploadException if a problem occurs during the transfer.

manage

boto.manage

boto.manage.cmdshell

boto.manage.propget

`boto.manage.propget.`**get**(*prop*, *choices=None*)

boto.manage.server

High-level abstraction of an EC2 server

class `boto.manage.server.`**Bundler**(*server*, *uname='root'*)

bundle(*bucket=None*, *prefix=None*, *key_file=None*, *cert_file=None*, *size=None*, *ssh_key=None*, *fp=None*, *clear_history=True*)

bundle_image(*prefix*, *size*, *ssh_key*)

copy_x509 (*key_file*, *cert_file*)

upload_bundle (*bucket*, *prefix*, *ssh_key*)

class boto.manage.server.**CommandLineGetter**

get (*cls*, *params*)

get_ami_id (*params*)

get_ami_list ()

get_description (*params*)

get_group (*params*)

get_instance_type (*params*)

get_key (*params*)

get_name (*params*)

get_quantity (*params*)

get_region (*params*)

get_zone (*params*)

class boto.manage.server.**Server** (*id=None*, ***kw*)

classmethod add_credentials (*cfg*, *aws_access_key_id*, *aws_secret_access_key*)

ami_id = None

console_output = None

classmethod create (*config_file=None*, *logical_volume=None*, *cfg=None*, ***params*)

Create a new instance based on the specified configuration file or the specified configuration and the passed in parameters.

If the config_file argument is not None, the configuration is read from there. Otherwise, the cfg argument is used.

The config file may include other config files with a #import reference. The included config files must reside in the same directory as the specified file.

The logical_volume argument, if supplied, will be used to get the current physical volume ID and use that as an override of the value specified in the config file. This may be useful for debugging purposes when you want to debug with a production config file but a test Volume.

The dictionary argument may be used to override any EC2 configuration values in the config file.

classmethod create_from_current_instances ()

classmethod create_from_instance_id (*instance_id*, *name*, *description=''*)

delete ()

description = None

elastic_ip = None

get_bundler (*uname='root'*)

get_cmdshell ()

get_ssh_client (*uname='root'*, *ssh_pwd=None*)

get_ssh_key_file()

groups = None

hostname = None

install(*pkg*)

instance_id = None

instance_type = None

key_name = None

launch_time = None

name = None

packages = []

plugins = []

private_hostname = None

production = None

put()

reboot()

region_name = None

reset_cmdshell()

run(*command*)

security_group = None

status = None

stop()

terminate()

wait()

zone = None

boto.manage.task

class boto.manage.task.**Task**(*id=None*, ***kw*)

A scheduled, repeating task that can be executed by any participating servers. The scheduling is similar to cron jobs. Each task has an hour attribute. The allowable values for hour are [0-23|*].

To keep the operation reasonably efficient and not cause excessive polling, the minimum granularity of a Task is hourly. Some examples:

hour='*' - the task would be executed each hour hour='3' - the task would be executed at 3AM GMT each day.

check()

Determine how long until the next scheduled time for a Task. Returns the number of seconds until the next scheduled time or zero if the task needs to be run immediately. If it's an hourly task and it's never been run, run it now. If it's a daily task and it's never been run and the hour is right, run it now.

command = None

hour = None

last_executed = None

last_output = None

last_status = None

message_id = None

name = None

run (*msg*, *vtimeout=60*)

start (*queue_name*)

classmethod **start_all** (*queue_name*)

class boto.manage.task.**TaskPoller** (*queue_name*)

poll (*wait=60*, *vtimeout=60*)

boto.manage.task.**check_hour** (*val*)

boto.manage.volume

class boto.manage.volume.**CommandLineGetter**

get (*cls*, *params*)

get_device (*params*)

get_mount_point (*params*)

get_name (*params*)

get_region (*params*)

get_size (*params*)

get_zone (*params*)

class boto.manage.volume.**Volume** (*id=None*, ***kw*)

archive ()

attach (*server=None*)

attachment_state = None

checkfs (*use_cmd=None*)

copy (*snapshot*)

classmethod **create** (***params*)

create_from_latest_snapshot (*name*, *size=None*)

create_from_snapshot (*name*, *snapshot*, *size=None*)

classmethod **create_from_volume_id** (*region_name*, *volume_id*, *name*)

delete (*delete_ebs_volume=False*)

detach (*force=False*)

device = None

format()

freeze()

get_ec2_connection()

get_snapshot_from_date(*date*)

get_snapshot_range(*snaps*, *start_date=None*, *end_date=None*)

get_snapshots()
Returns a list of all completed snapshots for this volume ID.

grow(*size*)

install_xfs()

make_ready(*server*)

mount()

mount_point = None

name = None

past_volume_ids = None

region_name = None

server = None

size = None

snapshot()

trim_snapshots(*delete=False*)
Trim the number of snapshots for this volume. This method always keeps the oldest snapshot. It then uses the parameters passed in to determine how many others should be kept.

The algorithm is to keep all snapshots from the current day. Then it will keep the first snapshot of the day for the previous seven days. Then, it will keep the first snapshot of the week for the previous four weeks. After than, it will keep the first snapshot of the month for as many months as there are.

unfreeze()

volume_id = None

volume_state = None

wait()

zone_name = None

pyami

boto.pyami

boto.pyami.bootstrap

class boto.pyami.bootstrap.**Bootstrap**
The Bootstrap class is instantiated and run as part of the PyAMI instance initialization process. The methods in this class will be run from the rc.local script of the instance and will be run as the root user.

The main purpose of this class is to make sure the boto distribution on the instance is the one required.

create_working_dir()

fetch_s3_file(*s3_file*)

load_boto()

load_packages()

main()

write_metadata()

boto.pyami.config

class boto.pyami.config.**Config**(*path=None*, *fp=None*, *do_load=True*)

dump()

dump_safe(*fp=None*)

dump_to_sdb(*domain_name*, *item_name*)

get(*section*, *name*, *default=None*)

get_instance(*name*, *default=None*)

get_user(*name*, *default=None*)

get_value(*section*, *name*, *default=None*)

getbool(*section*, *name*, *default=False*)

getfloat(*section*, *name*, *default=0.0*)

getint(*section*, *name*, *default=0*)

getint_user(*name*, *default=0*)

load_credential_file(*path*)
Load a credential file as is setup like the Java utilities

load_from_path(*path*)

load_from_sdb(*domain_name*, *item_name*)

save_option(*path*, *section*, *option*, *value*)
Write the specified Section.Option to the config file specified by path. Replace any previous value. If the path doesn't exist, create it. Also add the option the the in-memory config.

save_system_option(*section*, *option*, *value*)

save_user_option(*section*, *option*, *value*)

setbool(*section*, *name*, *value*)

boto.pyami.copybot

class boto.pyami.copybot.**CopyBot**

copy_bucket_acl()

copy_key_acl (*src*, *dst*)

copy_keys ()

copy_log ()

main ()

boto.pyami.installers

class boto.pyami.installers.**Installer** (*config_file=None*)
Abstract base class for installers

add_cron (*name*, *minute*, *hour*, *mday*, *month*, *wday*, *who*, *command*, *env=None*)
Add an entry to the system crontab.

add_env (*key*, *value*)
Add an environemnt variable

add_init_script (*file*)
Add this file to the init.d directory

install ()
Do whatever is necessary to "install" the package.

start (*service_name*)
Start a service.

stop (*service_name*)
Stop a service.

boto.pyami.installers.ubuntu

boto.pyami.installers.ubuntu.apache

class boto.pyami.installers.ubuntu.apache.**Apache** (*config_file=None*)
Install apache2, mod_python, and libapache2-svn

install ()

main ()

boto.pyami.installers.ubuntu.ebs

Automated installer to attach, format and mount an EBS volume. This installer assumes that you want the volume formatted as an XFS file system. To drive this installer, you need the following section in the boto config passed to the new instance. You also need to install dateutil by listing python-dateutil in the list of packages to be installed in the Pyami seciont of your boto config file.

If there is already a device mounted at the specified mount point, the installer assumes that it is the ephemeral drive and unmounts it, remounts it as /tmp and chmods it to 777.

Config file section:

```
[EBS]
volume_id = <the id of the EBS volume, should look like vol-xxxxxxxx>
logical_volume_name = <the name of the logical volume that contaings
    a reference to the physical volume to be mounted. If this parameter
    is supplied, it overrides the volume_id setting.>
```

```
device = <the linux device the EBS volume should be mounted on>
mount_point = <directory to mount device, defaults to /ebs>
```

class `boto.pyami.installers.ubuntu.ebs.`**`EBSInstaller`**(*config_file=None*)

Set up the EBS stuff

`attach`()

`create_backup_cleanup_script`(*use_tag_based_cleanup=False*)

`create_backup_script`()

`handle_mount_point`()

`install`()

`main`()

`make_fs`()

`update_fstab`()

boto.pyami.installers.ubuntu.installer

class `boto.pyami.installers.ubuntu.installer.`**`Installer`**(*config_file=None*)

Base Installer class for Ubuntu-based AMI's

`add_cron`(*name*, *command*, *minute='*'*, *hour='*'*, *mday='*'*, *month='*'*, *wday='*'*, *who='root'*, *env=None*)

Write a file to /etc/cron.d to schedule a command env is a dict containing environment variables you want to set in the file name will be used as the name of the file

`add_env`(*key*, *value*)

Add an environemnt variable For Ubuntu, the best place is /etc/environment. Values placed here do not need to be exported.

`add_init_script`(*file*, *name*)

Add this file to the init.d directory

`create_user`(*user*)

Create a user on the local system

`install`()

This is the only method you need to override

`start`(*service_name*)

`stop`(*service_name*)

boto.pyami.installers.ubuntu.mysql

This installer will install mysql-server on an Ubuntu machine. In addition to the normal installation done by apt-get, it will also configure the new MySQL server to store it's data files in a different location. By default, this is /mnt but that can be configured in the [MySQL] section of the boto config file passed to the instance.

class `boto.pyami.installers.ubuntu.mysql.`**`MySQL`**(*config_file=None*)

`change_data_dir`(*password=None*)

`install`()

main()

boto.pyami.installers.ubuntu.trac

class `boto.pyami.installers.ubuntu.trac.`**Trac**(*config_file=None*)

Install Trac and DAV-SVN Sets up a Vhost pointing to [Trac]->home Using the config parameter [Trac]->hostname Sets up a trac environment for every directory found under [Trac]->data_dir

[Trac] name = My Foo Server hostname = trac.foo.com home = /mnt/sites/trac data_dir = /mnt/trac svn_dir = /mnt/subversion server_admin = root@foo.com sdb_auth_domain = users # Optional SSLCertificateFile = /mnt/ssl/foo.crt SSLCertificateKeyFile = /mnt/ssl/foo.key SSLCertificateChainFile = /mnt/ssl/FooCA.crt

install()

main()

setup_vhost()

boto.pyami.launch_ami

`boto.pyami.launch_ami.`**main**()

`boto.pyami.launch_ami.`**usage**()

boto.pyami.scriptbase

class `boto.pyami.scriptbase.`**ScriptBase**(*config_file=None*)

main()

mkdir(*path*)

notify(*subject*, *body=''*)

run(*command*, *notify=True*, *exit_on_error=False*, *cwd=None*)

umount(*path*)

boto.pyami.startup

class `boto.pyami.startup.`**Startup**(*config_file=None*)

main()

run_scripts()

services

boto.services

boto.services.bs

class `boto.services.bs.`**BS**

Commands = {'reset': 'Clear input queue and output bucket', 'status': 'Report on the status of the service buckets and qu

Usage = 'usage: %prog [options] config_file command'

do_batches()

do_reset()

do_retrieve()

do_start()

do_status()

do_submit()

main()

print_command_help()

boto.services.message

class boto.services.message.**ServiceMessage**(*queue=None*, *body=None*, *xml_attrs=None*)

for_key(*key*, *params=None*, *bucket_name=None*)

boto.services.result

class boto.services.result.**ResultProcessor**(*batch_name*, *sd*, *mimetype_files=None*)

LogFileName = 'log.csv'

calculate_stats(*msg*)

get_results(*path*, *get_file=True*, *delete_msg=True*)

get_results_from_bucket(*path*)

get_results_from_domain(*path*, *get_file=True*)

get_results_from_queue(*path*, *get_file=True*, *delete_msg=True*)

log_message(*msg*, *path*)

process_record(*record*, *path*, *get_file=True*)

boto.services.service

class boto.services.service.**Service**(*config_file=None*, *mimetype_files=None*)

ProcessingTime = 60

cleanup()

delete_message(*message*)

get_file(*message*)

main(*notify=False*)

process_file(*in_file_name*, *msg*)

put_file (*bucket_name*, *file_path*, *key_name=None*)

read_message ()

save_results (*results*, *input_message*, *output_message*)

shutdown ()

split_key (*key*)

write_message (*message*)

boto.services.servicedef

class `boto.services.servicedef.`**ServiceDef** (*config_file*, *aws_access_key_id=None*, *aws_secret_access_key=None*)

get (*name*, *default=None*)

get_obj (*name*)

Returns the AWS object associated with a given option.

The heuristics used are a bit lame. If the option name contains the word 'bucket' it is assumed to be an S3 bucket, if the name contains the word 'queue' it is assumed to be an SQS queue and if it contains the word 'domain' it is assumed to be a SimpleDB domain. If the option name specified does not exist in the config file or if the AWS object cannot be retrieved this returns None.

getbool (*option*, *default=False*)

getint (*option*, *default=0*)

has_option (*option*)

boto.services.sonofmmm

class `boto.services.sonofmmm.`**SonOfMMM** (*config_file=None*)

process_file (*in_file_name*, *msg*)

queue_files ()

shutdown ()

boto.services.submit

class `boto.services.submit.`**Submitter** (*sd*)

get_key_name (*fullpath*, *prefix*)

submit_file (*path*, *metadata=None*, *cb=None*, *num_cb=0*, *prefix='/'*)

submit_path (*path*, *tags=None*, *ignore_dirs=None*, *cb=None*, *num_cb=0*, *status=False*, *prefix='/'*)

write_message (*key*, *metadata*)

STS

boto.sts

`boto.sts.`**`connect_to_region`**(*region_name*, ***kw_params*)

Given a valid region name, return a `boto.sts.connection.STSConnection`.

Type str

Parameters **`region_name`** – The name of the region to connect to.

Return type `boto.sts.connection.STSConnection` or `None`

Returns A connection to the given region, or None if an invalid region name is given

`boto.sts.`**`get_region`**(*region_name*, ***kw_params*)

Find and return a `boto.regioninfo.RegionInfo` object given a region name.

Type str

Param The name of the region.

Return type `boto.regioninfo.RegionInfo`

Returns The RegionInfo object for the given region or None if an invalid region name is provided.

`boto.sts.`**`regions`**()

Get all available regions for the STS service.

Return type *list*

Returns A list of `boto.regioninfo.RegionInfo` instances

class `boto.sts.`**`STSConnection`**(*aws_access_key_id=None*, *aws_secret_access_key=None*, *is_secure=True*, *port=None*, *proxy=None*, *proxy_port=None*, *proxy_user=None*, *proxy_pass=None*, *debug=0*, *https_connection_factory=None*, *region=None*, *path='/'*, *converter=None*)

`APIVersion` = **'2011-06-15'**

`DefaultRegionEndpoint` = **'sts.amazonaws.com'**

`DefaultRegionName` = **'us-east-1'**

`get_federation_token`(*name*, *duration=None*, *policy=None*)

Parameters

- **`name`** (*`str`*) – The name of the Federated user associated with the credentials.
- **`duration`** (*`int`*) – The number of seconds the credentials should remain valid.
- **`policy`** (*`str`*) – A JSON policy to associate with these credentials.

`get_session_token`(*duration=None*, *force_new=False*)

Return a valid session token. Because retrieving new tokens from the Secure Token Service is a fairly heavyweight operation this module caches previously retrieved tokens and returns them when appropriate. Each token is cached with a key consisting of the region name of the STS endpoint concatenated with the requesting user's access id. If there is a token in the cache meeting with this key, the session expiration is checked to make sure it is still valid and if so, the cached token is returned. Otherwise, a new session token is requested from STS and it is placed into the cache and returned.

Parameters

- **`duration`** (*`int`*) – The number of seconds the credentials should remain valid.
- **`force_new`** (*`bool`*) – If this parameter is True, a new session token will be retrieved from the Secure Token Service regardless of whether there is a valid cached token or not.

boto.sts.credentials

class `boto.sts.credentials.`**`Credentials`**(*parent=None*)

Variables

- ***`access_key`*** – The AccessKeyID.
- ***`secret_key`*** – The SecretAccessKey.
- **`session_token`** – The session token that must be passed with requests to use the temporary credentials
- **`expiration`** – The timestamp for when the credentials will expire

`endElement`(*name*, *value*, *connection*)

classmethod `from_json`(*json_doc*)

Create and return a new Session Token based on the contents of a JSON document.

Parameters **`json_doc`** (*`str`*) – A string containing a JSON document with a previously saved Credentials object.

`is_expired`(*time_offset_seconds=0*)

Checks to see if the Session Token is expired or not. By default it will check to see if the Session Token is expired as of the moment the method is called. However, you can supply an optional parameter which is the number of seconds of offset into the future for the check. For example, if you supply a value of 5, this method will return a True if the Session Token will be expired 5 seconds from this moment.

Parameters **`time_offset_seconds`** (*`int`*) – The number of seconds into the future to test the Session Token for expiration.

classmethod `load`(*file_path*)

Create and return a new Session Token based on the contents of a previously saved JSON-format file.

Parameters **`file_path`** (*`str`*) – The fully qualified path to the JSON-format file containing the previously saved Session Token information.

`save`(*file_path*)

Persist a Session Token to a file in JSON format.

Parameters **`path`** (*`str`*) – The fully qualified path to the file where the the Session Token data should be written. Any previous data in the file will be overwritten. To help protect the credentials contained in the file, the permissions of the file will be set to readable/writable by owner only.

`startElement`(*name*, *attrs*, *connection*)

`to_dict`()

Return a Python dict containing the important information about this Session Token.

class `boto.sts.credentials.`**`FederationToken`**(*parent=None*)

Variables

- ***`credentials`*** – A Credentials object containing the credentials.
- **`federated_user_arn`** – ARN specifying federated user using credentials.

- **federated_user_id** – The ID of the federated user using credentials.
- **packed_policy_size** – A percentage value indicating the size of the policy in packed form

endElement (*name*, *value*, *connection*)

startElement (*name*, *attrs*, *connection*)

About the Documentation

boto's documentation uses the Sphinx documentation system, which in turn is based on docutils. The basic idea is that lightly-formatted plain-text documentation is transformed into HTML, PDF, and any other output format.

To actually build the documentation locally, you'll currently need to install Sphinx – `easy_install Sphinx` should do the trick.

Then, building the html is easy; just `make html` from the `docs` directory.

To get started contributing, you'll want to read the ReStructuredText Primer. After that, you'll want to read about the Sphinx-specific markup that's used to manage metadata, indexing, and cross-references.

The main thing to keep in mind as you write and edit docs is that the more semantic markup you can add the better. So:

```
Import ``boto`` to your script...
```

Isn't nearly as helpful as:

```
Add :mod:`boto` to your script...
```

This is because Sphinx will generate a proper link for the latter, which greatly helps readers. There's basically no limit to the amount of useful markup you can add.

The fabfile

There is a Fabric file that can be used to build and deploy the documentation to a webserver that you ssh access to.

To build and deploy:

```
cd docs/
fab deploy:remote_path='/var/www/folder/whatever' --hosts=user@host
```

This will get the latest code from subversion, add the revision number to the docs conf.py file, call `make html` to build the documentation, then it will tarball it up and scp up to the host you specified and untarball it in the folder you specified creating a symbolic link from the untarballed versioned folder to `{remote_path}/boto-docs`.

Indices and tables

- genindex
- modindex
- search

Python Module Index

Index