# JAVA S3 EXAMPLES

## SETUP

The following examples may require some or all of the following java classes to be imported:

```
import java.io.ByteArrayInputStream;
import java.io.File;
import java.util.List;
import com.amazonaws.auth.AWSCredentials;
import com.amazonaws.auth.BasicAWSCredentials;
import com.amazonaws.util.StringUtils;
import com.amazonaws.services.s3.AmazonS3;
import com.amazonaws.services.s3.AmazonS3Client;
import com.amazonaws.services.s3.model.Bucket;
import com.amazonaws.services.s3.model.CannedAccessControlList;
import com.amazonaws.services.s3.model.GeneratePresignedUrlRequest;
import com.amazonaws.services.s3.model.GetObjectRequest;
import com.amazonaws.services.s3.model.ObjectListing;
import com.amazonaws.services.s3.model.ObjectMetadata;
import com.amazonaws.services.s3.model.S3ObjectSummary;
```

## CREATING A CONNECTION

This creates a connection so that you can interact with the server.

```
String accessKey = "insert your access key here!";
String secretKey = "insert your secret key here!";

AWSCredentials credentials = new BasicAWSCredentials(accessKey, secretKey);
AmazonS3 conn = new AmazonS3Client(credentials);
conn.setEndpoint("objects.dreamhost.com");
```

## LISTING OWNED BUCKETS

This gets a list of Buckets that you own. This also prints out the bucket name and creation date of each bucket.

```
List<Bucket> buckets = conn.listBuckets();
for (Bucket bucket : buckets) {
        System.out.println(bucket.getName() + "\t" +
                StringUtils.fromDate(bucket.getCreationDate()));
}
```

The output will look something like this:

```
mahbuckat1   2011-04-21T18:05:39.000Z
mahbuckat2   2011-04-21T18:05:48.000Z
mahbuckat3   2011-04-21T18:07:18.000Z
```

## CREATING A BUCKET

This creates a new bucket called `my-new-bucket`

```
Bucket bucket = conn.createBucket("my-new-bucket");
```

## LISTING A BUCKET'S CONTENT

This gets a list of objects in the bucket. This also prints out each object's name, the file size, and last modified date.

```
ObjectListing objects = conn.listObjects(bucket.getName());
do {
        for (S3ObjectSummary objectSummary : objects.getObjectSummaries()) {
                System.out.println(objectSummary.getKey() + "\t" +
                        ObjectSummary.getSize() + "\t" +
                        StringUtils.fromDate(objectSummary.getLastModified()));
        }
        objects = conn.listNextBatchOfObjects(objects);
} while (objects.isTruncated());
```

The output will look something like this:

```
myphoto1.jpg 251262  2011-08-08T21:35:48.000Z
myphoto2.jpg 262518  2011-08-08T21:38:01.000Z
```

## DELETING A BUCKET

**Note:** The Bucket must be empty! Otherwise it won't work!

```
conn.deleteBucket(bucket.getName());
```

## FORCED DELETE FOR NON-EMPTY BUCKETS

**Attention:** not available

## CREATING AN OBJECT

This creates a file `hello.txt` with the string `"Hello World!"`

```
ByteArrayInputStream input = new ByteArrayInputStream("Hello World!".getBytes());
conn.putObject(bucket.getName(), "hello.txt", input, new ObjectMetadata());
```

## CHANGE AN OBJECT'S ACL

This makes the object `hello.txt` to be publicly readable, and `secret_plans.txt` to be private.

```
conn.setObjectAcl(bucket.getName(), "hello.txt", CannedAccessControlList.PublicRead);
conn.setObjectAcl(bucket.getName(), "secret_plans.txt", CannedAccessControlList.Private);
```

## DOWNLOAD AN OBJECT (TO A FILE)

This downloads the object `perl_poetry.pdf` and saves it in `/home/larry/documents`

```
conn.getObject(
        new GetObjectRequest(bucket.getName(), "perl_poetry.pdf"),
        new File("/home/larry/documents/perl_poetry.pdf")
);
```

## DELETE AN OBJECT

This deletes the object `goodbye.txt`

```
conn.deleteObject(bucket.getName(), "goodbye.txt");
```

## GENERATE OBJECT DOWNLOAD URLS (SIGNED AND UNSIGNED)

This generates an unsigned download URL for `hello.txt`. This works because we made `hello.txt` public by setting the ACL above. This then generates a signed download URL for `secret_plans.txt` that will work for 1 hour. Signed download URLs will work for the time period even if the object is private (when the time period is up, the URL will stop working).

**Note:** The java library does not have a method for generating unsigned URLs, so the example below just generates a signed URL.

```
GeneratePresignedUrlRequest request = new GeneratePresignedUrlRequest(bucket.getName(), "secr
System.out.println(conn.generatePresignedUrl(request));
```

The output will look something like this:

```
https://my-bucket-name.objects.dreamhost.com/secret_plans.txt?Signature=XXXXXXXXXXXXXXXXXXXXX
```