PHP S3 EXAMPLES

CREATING A CONNECTION

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

```php
<?php
define('AWS_KEY', 'place access key here');
define('AWS_SECRET_KEY', 'place secret key here');
define('AWS_CANONICAL_ID', 'your DHO Username');
define('AWS_CANONICAL_NAME', 'Also your DHO Username!');
$HOST = 'objects.dreamhost.com';

// require the amazon sdk for php library
require_once 'AWSSDKforPHP/sdk.class.php';

// Instantiate the S3 class and point it at the desired host
$Connection = new AmazonS3(array(
        'key' => AWS_KEY,
        'secret' => AWS_SECRET_KEY,
        'canonical_id' => AWS_CANONICAL_ID,
        'canonical_name' => AWS_CANONICAL_NAME,
));
$Connection->set_hostname($HOST);
$Connection->allow_hostname_override(false);

// Set the S3 class to use objects.dreamhost.com/bucket
// instead of bucket.objects.dreamhost.com
$Connection->enable_path_style();
```

LISTING OWNED BUCKETS

This gets a list of CFSimpleXML objects representing buckets that you own. This also prints out the bucket name and creation date of each bucket.

```php
<?php
$ListResponse = $Connection->list_buckets();
$Buckets = $ListResponse->body->Buckets->Bucket;
foreach ($Buckets as $Bucket) {
        echo $Bucket->Name . "\t" . $Bucket->CreationDate . "\n";
}
```

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` and returns a `CFResponse` object.

Note: This command requires a region as the second argument, so we use `AmazonS3::REGION_US_E1`, because this constant is `''`

```php
<?php
$Connection->create_bucket('my-new-bucket', AmazonS3::REGION_US_E1);
```

LIST A BUCKET'S CONTENT

This gets an array of `CFSimpleXML` objects representing the objects in the bucket. This then prints out each object's name, the file size, and last modified date.

```
<?php
$ObjectsListResponse = $Connection->list_objects($bucketname);
$Objects = $ObjectsListResponse->body->Contents;
foreach ($Objects as $Object) {
        echo $Object->Key . "\t" . $Object->Size . "\t" . $Object->LastModified . "\n";
}
```

Note: If there are more than 1000 objects in this bucket, you need to check $ObjectListResponse->body->isTruncated and run again with the name of the last key listed. Keep doing this until isTruncated is not true.

The output will look something like this if the bucket has some files:

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

DELETING A BUCKET

This deletes the bucket called `my-old-bucket` and returns a `CFResponse` object

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

```
<?php
$Connection->delete_bucket('my-old-bucket');
```

FORCED DELTE FOR NON-EMPTY BUCKETS

This will delete the bucket even if it is not empty.

```
<?php
$Connection->delete_bucket('my-old-bucket', 1);
```

CREATING AN OBJECT

This creates an object `hello.txt` with the string `"Hello World!"`

```
<?php
$Connection->create_object('my-bucket-name', 'hello.txt', array(
        'body' => "Hello World!",
));
```

CHANGE AN OBJECT'S ACL

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

```
<?php
$Connection->set_object_acl('my-bucket-name', 'hello.txt', AmazonS3::ACL_PUBLIC);
$Connection->set_object_acl('my-bucket-name', 'secret_plans.txt', AmazonS3::ACL_PRIVATE);
```

DELETE AN OBJECT

This deletes the object `goodbye.txt`

```
<?php
$Connection->delete_object('my-bucket-name', 'goodbye.txt');
```

DOWNLOAD AN OBJECT (TO A FILE)

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

```
<?php
$FileHandle = fopen('/home/larry/documents/poetry.pdf', 'w+');
$Connection->get_object('my-bucket-name', 'poetry.pdf', array(
        'fileDownload' => $FileHandle,
));
```

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).

```
<?php
my $plans_url = $Connection->get_object_url('my-bucket-name', 'hello.txt');
echo $plans_url . "\n";
my $secret_url = $Connection->get_object_url('my-bucket-name', 'secret_plans.txt', '1 hour');
echo $secret_url . "\n";
```

The output of this will look something like:

```
http://objects.dreamhost.com/my-bucket-name/hello.txt
http://objects.dreamhost.com/my-bucket-name/secret_plans.txt?Signature=XXXXXXXXXXXXXXXXXXXXXXX
```