CHAPTER 16

Using Online Storage Services

Amazon S3 and comparable services are intriguing players to recently enter the world of online storage. As we produce more digital content to share, we often need to store that content in a place accessible to others. Moreover, if you are building a service for others to use and need to have access to lots of storage, it's valuable to be able to scale that storage up quickly without lots of upfront capital investment in storage hardware.

Amazon S3 is the poster child in the arena of online storage—and hence is the primary focus of this chapter. I will also cover some other web sites that have similar offerings but with important twists. For instance, some of these services are meant to be used as backup services and not really for serving up digital objects for web applications.

Some of the online storage services have APIs, which makes them highly mashable. They include the following:

* Amazon S3 (`http://aws.amazon.com/s3`)
* Box.net (`http://box.net/`) with its API[1]
* MediaMax (`http://www.mediamax.com/`) with its API[2]
* Omnidrive (`http://www.omnidrive.com/`) and its API[3]

This chapter shows the basics of using the most well known of the online storage services: Amazon S3.

Introducing Amazon S3

Amazon S3 (the S3 stands for Simple Storage Service) is described in the following way:[4]

> *Amazon S3 provides a simple web services interface that can be used to store and retrieve any amount of data, at any time, from anywhere on the Web. It gives any developer access to the same highly scalable, reliable, fast, inexpensive data storage infrastructure that Amazon uses to run its own global network of web sites.*

There is no direct user interface for S3; it is meant as a technical infrastructure upon which developers can build services. The only interface to S3 provided by Amazon.com is a web services API. You can access S3 through its REST or SOAP interface directly or via third-party language-specific API kits that use the REST or SOAP interface. Using the API kits generally makes accessing S3 easier, provided they are well documented and cover the parts of the API that you care about. Sometimes they do not shield you from all the subtleties of the underlying API. For instance, to use the SOAP/WSDL interface to S3, you need to understand how to sign your calls, a topic I will cover briefly in this chapter.

In the following sections, I will guide you through how to get started with Amazon S3, outlining how to use the API and referring you to the API's detailed documentation as appropriate.

1. http://enabled.box.net/

2. http://www.mediamax.com/webservices/

3. http://dev.omnidrive.com/HomePage

4. http://aws.amazon.com/s3, which redirects to http://www.amazon.com/gp/browse.html?node=16427261

Rationale for S3

Why use Amazon S3? Here are some arguments for using S3:

* S3 is potentially cheaper than the alternative solutions. For instance, Don MacAskill, of SmugMug, a photo-hosting service akin to Flickr, estimated that SmugMug saved about $692,000 in 2006 by using S3 instead of buying and maintaining the equivalent amount of storage.[5] (Note that the comparisons might have changed with Amazon.com's new pricing model.[6])
* S3 promises high scalability, both in volume and in the rate of change of storage needs. You "pay as you go" and pay for what you use. That means you don't have to invest up front in buying the maximum amount of storage you think you will need. The utility model lowers the barrier of entry to the level that a relatively poor individual can afford to create a Web 2.0 application.
* Robustness is part of the picture since Amazon.com claims it runs its own infrastructure on S3, and therefore you can expect reliability similar to that of Amazon.com itself.[7]

For specific examples of how S3 is being used, see the following:

* `http://jeremy.zawodny.com/blog/archives/007641.html` for a list of backup services and tools that use S3
* `http://solutions.amazonwebservices.com/connect/kbcategory.jspa?categoryID=66` to see the solutions that developers have already built

Because online storage is accessed through the Internet, latency (and how it will affect your application) is an important factor to consider when looking at S3 and similar services.

5. http://blogs.smugmug.com/don/files/ETech-SmugMug-Amazon-2007.pdf

6. http://blogs.smugmug.com/don/2007/05/01/amazon-s3-new-pricing-model/

7. http://www.amazon.com/S3-FAQs-AWS-home-page/b/ref=sc_fe_c_0_16427261_9?&node=16427271&no=16427261#as9

Conceptual Structure of Amazon S3

To get started, read the core concepts documented here:

`http://docs.amazonwebservices.com/AmazonS3/2006-03-01/CoreConcepts.html`

At its heart, S3 is conceptually simple; it lets you store objects in *buckets*. An object is associated with a bucket via a key. There are authentication and authorization schemes associated with S3 to grant you control over access to the buckets and objects. You can associate some amount of metadata (in the form of key-value pairs) with objects.

The following are a few more important points:

* Since a bucket name is global across the S3 service (akin to a domain name), each developer account can have up to 100 buckets at any one time. Bucket names can contain only alphanumeric characters, underscores (_), periods (.), and hyphens (-). They must be between 3 and 255 characters long, and buckets with names containing uppercase characters are not accessible using the virtual hosting method.[8]
* Objects consist of object data and associated metadata. An object can hold up to 5 gigabytes of data.[9]
* A key is like a filename for an object and must be unique within a bucket. Its UTF-8 encoding must be at most 1,024 bytes long.
* You use prefixes and delimiters in keys to simulate a hierarchical (folder within folder-like) organization within buckets.[10] (Buckets cannot contain other buckets.)
* For both REST and SOAP requests to S3, the user metadata size associated with objects is limited to 2,000 bytes. They are structured as key-value pairs.[11]
* There is an authentication and authorization system in place. You can have fine-grained authorization, where you can associate permissions with specific users or with larger preset groups (the owner, everyone, or authenticated users). Permissions are read, write, or full control.[12]
* You can retrieve a `.torrent` file for any publicly available object by adding a `?torrent` query string parameter at the end of the REST `GET` request for the object.[13]
* There is virtual hosting of buckets that allows one to associate your own non-Amazon.com domain name with an S3 bucket.[14] For example, objects accessible at the following URL:

8. http://docs.amazonwebservices.com/AmazonS3/2006-03-01/UsingBucket.html
9. http://docs.amazonwebservices.com/AmazonS3/2006-03-01/UsingObjects.html
10. http://docs.amazonwebservices.com/AmazonS3/2006-03-01/ListingKeysHierarchy.html
11. http://docs.amazonwebservices.com/AmazonS3/2006-03-01/UsingMetadata.html
12. http://docs.amazonwebservices.com/AmazonS3/2006-03-01/UsingAccessControl.html
13. http://docs.amazonwebservices.com/AmazonS3/2006-03-01/S3TorrentRetrieve.html
14. http://docs.amazonwebservices.com/AmazonS3/2006-03-01/VirtualHosting.html

```
http://s3.amazonaws.com/{bucket}/{key}
```

are also accessible at the following (provided that the bucket name has no uppercase characters):

```
http://{bucket}.s3.amazonaws.com/{key}
```

For example, the following:

```
http://s3.amazonaws.com/raymondyee/858Xtoc___.pdf
```

is accessible here:

```
http://raymondyee.s3.amazonaws.com/858Xtoc___.pdf
```

SIGNING UP FOR AMAZON AWS

You need an Amazon Web Services (AWS) key and secret to use S3. The home page for AWS is here:

```
http://aws.amazon.com
```

You can find the documentation for the e-commerce services part of AWS here:

```
http://www.amazon.com/gp/browse.html?node=12738641
```

You need to sign up for an AWS account to get access keys:

```
http://www.amazon.com/gp/aws/registration/registration-form.html
```

To get your keys if you are already a member, go here:

```
http://aws-
portal.amazon.com/gp/aws/developer/account/index.html/?ie=UTF8&action=acces
s-key
```

After you get an access key ID and a secret access key, you can also use an X.509 certificate.

The Firefox S3 Extension Gets You Started with S3

As I have argued throughout the book, it's helpful to learn an application well before diving into its API. That S3 has no built-in user interface means you have to either program S3 yourself right from the start or use someone else's user interface. I recommend installing the S3 Firefox Explorer add-on to get a UI to not only manage your files on S3 but to also learn how S3 works. You can get the extension from here:

```
http://www.rjonna.com/ext/s3fox.php
```

The extension is a great learning tool for S3. Using it, for instance, you can quickly create a bucket and populate that bucket with an object. You can then test the code included in this chapter by reading the list of buckets and what is contained in them. Without a UI tool such as the Firefox extension, you would first have to get your code working to populate the buckets. Figure 16-1 shows the S3 Firefox Explorer add-on. The left panel is an explorer-like interface to your desktop. The right panel shows you your buckets and objects within the folders. You can edit the access control list (ACL) for each object. You can also copy the URL for an object.

Insert 858Xf1601.tif

Figure 16-1. S3 Firefox Explorer add-on

Similarly, you might want to install S3Drive (`http://www.s3drive.net/`) on Microsoft Windows to have fairly seamless integration in Windows. The S3Drive service makes S3 look like a partition on a local hard drive.

Using the S3 REST Interface

The S3 REST interface is truly RESTful—you think in terms of resources, such as services (to get a list of all your buckets), buckets, and objects—and they have standard methods. See the following for a list of resources and methods:

`http://docs.amazonwebservices.com/AmazonS3/2006-03-01/RESTAPI.html`

Using the REST interface is a bit tricky because of the following:

* How authentication is handled, specifically, how the signature is calculated[15]
* The use of authorization control lists to handle authorization[16]
* How metadata is implemented

In this section, I will show one specific, relatively simple `GET` example to demonstrate how to use the REST interface. Let's first use the query string request authentication alternative, which doesn't require the use of HTTP Authorization headers. As the documentation indicates, "The practice of signing a request and giving it to a third-party for execution is suitable only for simple object `GET` requests."

15. http://docs.amazonwebservices.com/AmazonS3/2006-03-01/RESTAuthentication.html

16. http://docs.amazonwebservices.com/AmazonS3/2006-03-01/RESTAccessPolicy.html

The REST endpoint is as follows:

`http://host.s3.amazonaws.com`

You need three query parameters:

* `AWSAccessKeyId`: Your access key
* `Expires`: When the signature expires, specified as the number of seconds since the epoch (00:00:00 UTC on January 1, 1970)
* `Signature`: The URL encoding of the Base64 encoding of the HMAC-SHA1 of `StringToSign` (defined in a moment)

I'll use the example data given in the documents and generate some Python and PHP code to demonstrate how to calculate the `Signature`. That is, I'll show you how to reproduce the results in the documentation. I'll use parameters (listed in Table 16-1) that draw from examples at the following location:

`http://docs.amazonwebservices.com/AmazonS3/2006-03-01/RESTAuthentication.html`

The parameters shown here would be used to access the object whose key is photos/puppy.jpg in the bucket named johnsmith. (Note that AWSAccessKeyId and AWSSecretAccessKey are not actually valid keys but are presented to illustrate the calculations.)

Table 16-1. The Values Used in This Example Calculation for S3 Parameters

Setting	Value
AWSAccessKeyId	0PN5J17HBGZHT7JJ3X82
AWSSecretAccessKey	uV3F3YluFJax1cknvbcGwgjvx4QpvB+leU8dUj2o
Expires	1175139620
Host	johnsmith.s3.amazonaws.com
Key	photos/puppy.jpg
HTTP-Verb	GET
Content-MD5	
Content-Type	
CanonicalizedAmzHeaders	
CanonicalizedResource	/johnsmith/photos/puppy.jpg

The pseudo-code for calculating the signature is (quoting from the documentation) as follows:

```
StringToSign = HTTP-VERB + "\n" +  Content-MD5 + "\n" +  Content-Type + "\n" +
  Expires + "\n" + CanonicalizedAmzHeaders + CanonicalizedResource;
Signature = URL-Encode( Base64( HMAC-SHA1( UTF-8-Encoding-Of( StringToSign ) )
));
```

We're told that the Signature based on the parameters in Table 16-1 should be rucSbH0yNEcP9oM2XNlouVI3BH4%3D. Let's figure out how we can reproduce this signature in Python and PHP.

First, here is the Python code to calculate the Signature:

```
    import sha, hmac, base64, urllib

    AWSAccessKeyId = "0PN5J17HBGZHT7JJ3X82"
    AWSSecretAccessKey = "uV3F3YluFJax1cknvbcGwgjvx4QpvB+leU8dUj2o"
    Expires = 1175139620
    HTTPVerb = "GET"
    ContentMD5 = ""
    ContentType = ""
    CanonicalizedAmzHeaders = ""
    CanonicalizedResource = "/johnsmith/photos/puppy.jpg"
    string_to_sign = HTTPVerb + "\n" +  ContentMD5 + "\n" +  ContentType + "\n"
+ ~CCC
      str(Expires) + "\n" + CanonicalizedAmzHeaders + CanonicalizedResource
    sig = base64.b64encode(~CCC
            hmac.new(AWSSecretAccessKey, string_to_sign, sha).digest())
    print urllib.urlencode({'Signature':sig})
```

This produces the following:

```
Signature=rucSbH0yNEcP9oM2XNlouVI3BH4%3D
```

Here's some corresponding PHP code to calculate the `Signature`:

```
<?php

# base64.encodestring
# The hex2b64 function is excerpted from the Amazon S3 PHP example library.
# http://developer.amazonwebservices.com/connect/entry.jspa?externalID=126

    function hex2b64($str) {
      $raw = '';
      for ($i=0; $i < strlen($str); $i+=2) {
        $raw .= chr(hexdec(substr($str, $i, 2)));
      }
      return base64_encode($raw);
    }

    require_once 'Crypt/HMAC.php';
    require_once 'HTTP/Request.php';

    $AWSAccessKeyId = "0PN5J17HBGZHT7JJ3X82";
    $AWSSecretAccessKey = "uV3F3YluFJax1cknvbcGwgjvx4QpvB+leU8dUj2o";
    $Expires = 1175139620;
    $HTTPVerb = "GET";
    $ContentMD5 = "";
    $ContentType = "";
    $CanonicalizedAmzHeaders = "";
    $CanonicalizedResource = "/johnsmith/photos/puppy.jpg";
    $string_to_sign = $HTTPVerb . "\n" . $ContentMD5 . "\n" . $ContentType .
"\n" .
      $Expires . "\n" . $CanonicalizedAmzHeaders . $CanonicalizedResource;

    $hasher =& new Crypt_HMAC($AWSSecretAccessKey, "sha1");
    $sig = hex2b64($hasher->hash($string_to_sign));
    echo 'Signature=',urlencode($sig);

?>
```

Note that this PHP code depends on two PEAR libraries that need to be installed:

* `Crypt_HMAC`[17]
* `HTTP_Request`[18]

Unfortunately, after you get those libraries installed, I can't recommend using the S3 sample code on a remote host, because it requires sending the secret over the wire. Run it on your own secure machine.

Once you have calculated the `Signature`, you can package the corresponding HTTP `GET` request:

```
http://johnsmith.s3.amazonaws.com/photos/puppy.jpg?AWSAccessKeyId=0PN5J17HBGZHT7
JJ3X
```

```
82&Signature=rucSbH0yNEcP9oM2XNlouVI3BH4%3D&Expires=1175139620
```

Listing Buckets Using the REST Interface

Now that you understand the basics behind signing an Amazon S3 request, I'll show you how to get a list of your S3 buckets. First I'll show the code, and then I'll offer an explanation:

```
def listBuckets(AWSAccessKeyId,AWSSecretAccessKey):
    """
    use the REST interface to get the list of buckets --
    without the use the Authorization HTTP header
    """
    import sha, hmac, base64, urllib
    import time
    # give an hour for the request to expire (3600s)
    expires = int(time.time()) + 3600
    string_to_sign = "GET\n\n\n%s\n/" % (expires)
    sig = base64.b64encode(~CCC
      hmac.new(AWSSecretAccessKey, string_to_sign, sha).digest())

    request = "http://s3.amazonaws.com?AWSAccessKeyId=%s&Expires=%s&%s" % \
              (AWSAccessKeyId, expires, urllib.urlencode({'Signature':sig}))
    return request


if __name__ == "__main__":
    AWSAccessKeyId='[AWSAccessKeyID]'
    AWSSecretAccessKey = '[SecretAccessKey]'
    print listBuckets(AWSAccessKeyId,AWSSecretAccessKey)
```

This code generates a URL of the following form that returns a list of buckets:

```
http://s3.amazonaws.com?AWSAccessKeyId={AWSAccessKeyId}&Expires=1196114919~CCC
&Signature={Signature}
```

Note how we use some of the same parameters as in the previous example (`AWSAccessKeyId`, `AWSSecretAccessKey`, and `Expires`) and calculate the `Signature` with the same combination of SHA-1 hashing and Base64 encoding.

17. http://pear.php.net/package/Crypt_HMAC

18. http://pear.php.net/package/HTTP_Request

Using the SOAP Interface to S3

The following code sample illustrates how to use the SOAP interface to S3:

```
import sha, hmac, base64, urllib

# list buckets for Amazon s3

AWSAccessKeyId='[AWSAccessKeyID]'
AWSSecretAccessKey = '[AWSSecretAccessKey]'
```

```
from SOAPpy import WSDL

import sha

def calcSig(key,text):
    import hmac, base64
    sig = base64.b64encode(hmac.new(key, text, sha).digest())
    return sig

def ListMyBuckets(s):
    from time import gmtime,strftime
    method = 'ListAllMyBuckets'
    ts = strftime("%Y-%m-%dT%H:%M:%S.000Z", gmtime())
    text = 'AmazonS3' + method + ts
    sig = calcSig(AWSSecretAccessKey,text)
    print "ListMyBuckets: ts,text,sig->", ts, text, sig
    return s.ListAllMyBuckets(AWSAccessKeyId=AWSAccessKeyId, ~CCC
Timestamp=ts,Signature=sig)

def CreateBucket(s, bucketName):
    from time import gmtime,strftime
    method = 'CreateBucket'
    print 'method: ', method
    ts = strftime("%Y-%m-%dT%H:%M:%S.000Z", gmtime())
    text = 'AmazonS3' + method + ts
    sig = calcSig(AWSSecretAccessKey,text)
    print "CreateBuckets: ts,text,sig->", ts, text, sig
    return s.CreateBucket(Bucket=bucketName, AWSAccessKeyId=AWSAccessKeyId,
~CCC
Timestamp=ts,Signature=sig)

if __name__ == '__main__':
    s = WSDL.Proxy("http://s3.amazonaws.com/doc/2006-03-01/AmazonS3.wsdl")
    print ListMyBuckets(s)
    CreateBucket(s,"test20071126RY")
    print ListMyBuckets(s)
```

You can learn the following about S3 from this code:

* As with the REST interface, you need to have an Amazon.com AWS access key ID and secret access key, which you can get if you sign up for an account at AWS (see Chapter 7 or the “Signing Up for Amazon AWS” sidebar earlier in this chapter for more details).
* Although S3 is accessible by the REST and SOAP interfaces, this code uses SOAP and WSDL. The WSDL for the service at the time of writing is at `http://s3.amazonaws.com/doc/2006-03-01/AmazonS3.wsdl`; you can get the location of the latest WSDL URL at `http://aws.amazon.com/s3`.

* Two methods are used in this code sample: `ListMyBuckets` and `CreateBuckets`. You can get the list of all the methods at `http://www.awszone.com/scratchpads/aws/s3.us/index.aws` (the technical documentation is at `http://developer.amazonwebservices.com/connect/kbcategory.jspa?categoryID=48`, which leads to `http://docs.amazonwebservices.com/AmazonS3/2006-03-01/`).
* Note that one of the complicated aspects is to calculate a signature, something you learned to do in the previous section.

Amazon S3 API Kits

In the following sections, you'll look at some libraries to S3 written in PHP and Python.

PHP

The following API kits are available:

* `php-aws`[19]
* `s3.class.zip` at Neurofuzzy.net, which looks like a popular class implementation[20]
* edoceo's `phps3tk`[21]

In this section, we'll concentrate on how to use `php-aws`. You can access the source using SVN. In your web browser, you can download the library from here:

```
http://php-aws.googlecode.com/svn/trunk/class.s3.php
```

You can find documentation for the S3 class here:

```
http://code.google.com/p/php-aws/wiki/S3Class
```

The following blog entry introduces `php-aws`:

```
http://sitening.com/blog/2007/01/30/introducing-php-aws/
```

Note the following about this library:

* Its use of `curl` means it is built to handle larger files.
* Only the public read or private ACL is currently implemented.
* There is no implementation of user metadata for objects.

To get started with `php-aws`, follow these steps:

1. Download `http://php-aws.googlecode.com/svn/trunk/class.s3.php` to your favorite local PHP directory. (In my case, this is `/home/rdhyee/phplib/php-aws/class.s3.php`.)
2. Try the following sample code to get you started (this code first lists your S3 buckets and then creates a bucket by the name of `mashupguidetest` if it doesn't already exist):

```
<?php
require_once("php-aws/class.s3.php");

$key = "[AWSAccessKeyID]";
$secret = "[SecretAccessKey]";

$s3 = new S3($key,$secret);

// get list of buckets
$buckets = $s3->getBuckets();
print_r($buckets);

// if the bucket "mashupguidetest" doesn't exist, create it
$BNAME = "mashupguidetest";
if (! $s3->bucketExists($BNAME)) {
  $s3->createBucket($BNAME);
}

// get list of buckets again
$buckets = $s3->getBuckets();
print_r($buckets);
```

Note For you to use `php-aws`, you need to have a command-line invokable instance of `curl` installed on your system. You might also need to set the `$_pathToCurl` parameter in `class.s3.php` so that `php-aws` can find `curl`.

---

19. http://code.google.com/p/php-aws/ and specifically http://php-aws.googlecode.com/svn/trunk/class.s3.php

20. http://neurofuzzy.net/2006/08/26/amazon-s3-php-class-update/ and http://www.neurofuzzy.net/wp-content/2006/03/s3.class.zip

21. http://www.edoceo.com/creo/phps3tk/

Python

Some Python-based S3 libraries are as follows:

* `boto`[22]
* HanzoiArchive's S3 `tools`[23]
* `BitBucket`[24]

22. http://code.google.com/p/boto/

23. http://www.hanzoarchives.com/development-projects/s3-tools/

24. http://cheeseshop.python.org/pypi/BitBucket/0.4a

I recommend looking at `boto` as a good choice of a library. One of the best ways to learn how to use `boto` is to read the tutorial here:

http://boto.googlecode.com/svn/trunk/doc/s3_tut.txt

You can learn the basics of using boto by studying the next code sample, which does the following:

* It reads the list of your S3 buckets and displays the name, creation date, and XML representation of the bucket's ACL.
* It reads the list of objects contained in a specific bucket, along with the last modified time stamp and the object's metadata.
* It uploads a file to a bucket and reads back the metadata of the newly uploaded file.

```
AWSAccessKeyId='[AWSAccessKeyId]'
AWSSecretAccessKey = '[AWSSecretAccessKey]'
FILENAME = 'D:\Document\PersonalInfoRemixBook\858Xtoc___.pdf'
BUCKET = 'mashupguidetest'

from boto.s3.connection import S3Connection

def upload_file(fname, bucket, key, acl='public-read', metadata=None):
    from boto.s3.key import Key

    fpic = Key(bucket)
    fpic.key = key
    #fpic.set_metadata('source','flickr')
    fpic.update_metadata(metadata)
    fpic.set_contents_from_filename(fname)
    fpic.set_acl(acl)
    return fpic

# set up a connection to S3

conn = S3Connection(AWSAccessKeyId, AWSSecretAccessKey)

# retrieve all the buckets
buckets = conn.get_all_buckets()
print "number of buckets:", len(buckets)

# print out the names, creation date, and the XML the represents the ACL
# of the bucket

for b in buckets:
    print "%s\t%s\t%s" % (b.name, b.creation_date, b.get_acl().acl.to_xml())

# get list of all files for the mashupguide bucket

print "keys in " + BUCKETmg_bucket = conn.get_bucket(BUCKET)
keys = mg_bucket.get_all_keys()
for key in keys:
    print "%s\t%s\t%s" % (key.name, key.last_modified, key.metadata)

# upload the table of contents to mashupguide bucket.
```

```
metadata = {'author':'Raymond Yee'}
upload_file(FILENAME,mg_bucket,'samplefile','public-read',metadata)

# read back the TOC
toc = mg_bucket.get_key('samplefile')
print toc.metadata
```

Summary

From reading this chapter, you should now know how to get started with the Amazon S3 API using PHP and Python. The APIs for other online storage systems are different but will have some conceptual similarity to S3.