

C++ S3 EXAMPLES

SETUP

The following contains includes and globals that will be used in later examples:

```
#include "libs3.h"
#include <stdlib.h>
#include <iostream>
#include <fstream>

const char access_key[] = "ACCESS_KEY";
const char secret_key[] = "SECRET_KEY";
const char host[] = "HOST";
const char sample_bucket[] = "sample_bucket";
const char sample_key[] = "hello.txt";
const char sample_file[] = "resource/hello.txt";

S3BucketContext bucketContext =
{
    host,
    sample_bucket,
    S3ProtocolHTTP,
    S3UriStylePath,
    access_key,
    secret_key
};

S3Status responsePropertiesCallback(
    const S3ResponseProperties *properties,
    void *callbackData)
{
    return S3StatusOK;
}

static void responseCompleteCallback(
    S3Status status,
    const S3ErrorDetails *error,
    void *callbackData)
{
    return;
}

S3ResponseHandler responseHandler =
{
    &responsePropertiesCallback,
    &responseCompleteCallback
};
```

CREATING (AND CLOSING) A CONNECTION

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

```
S3_initialize("s3", S3_INIT_ALL, host);
// Do stuff...
S3_deinitialize();
```

LISTING OWNED BUCKETS

This gets a list of Buckets that you own. This also prints out the bucket name, owner ID, and display name for each bucket.

```
static S3Status listServiceCallback(
    const char *ownerId,
```

```

        const char *ownerDisplayName,
        const char *bucketName,
        int64_t creationDate, void *callbackData)
{
    bool *header_printed = (bool*) callbackData;
    if (!*header_printed) {
        *header_printed = true;
        printf("%-22s", "      Bucket");
        printf("  %-20s  %-12s", "      Owner ID", "Display Name");
        printf("\n");
        printf("-----");
        printf("  -----" "  -----");
        printf("\n");
    }

    printf("%-22s", bucketName);
    printf("  %-20s  %-12s", ownerId ? ownerId : "", ownerDisplayName ? ownerDisplayName : "");
    printf("\n");

    return S3StatusOK;
}

S3ListServiceHandler listServiceHandler =
{
    responseHandler,
    &listServiceCallback
};
bool header_printed = false;
S3_list_service(S3ProtocolHTTP, access_key, secret_key, host, 0, &listServiceHandler, &header_printed);

```

CREATING A BUCKET

This creates a new bucket.

```
S3_create_bucket(S3ProtocolHTTP, access_key, secret_key, host, sample_bucket, S3CannedAclPriv
```

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.

```

static S3Status listBucketCallback(
    int isTruncated,
    const char *nextMarker,
    int contentsCount,
    const S3ListBucketContent *contents,
    int commonPrefixesCount,
    const char **commonPrefixes,
    void *callbackData)
{
    printf("%-22s", "      Object Name");
    printf("  %-5s  %-20s", "Size", "Last Modified");
    printf("\n");
    printf("-----");
    printf("  -----" "  -----");
    printf("\n");

    for (int i = 0; i < contentsCount; i++) {
        char timebuf[256];
        char sizebuf[16];
        const S3ListBucketContent *content = &(contents[i]);
        time_t t = (time_t) content->lastModified;

        strftime(timebuf, sizeof(timebuf), "%Y-%m-%dT%H:%M:%SZ", gmtime(&t));
        sprintf(sizebuf, "%5llu", (unsigned long long) content->size);
        printf("%-22s  %s  %s\n", content->key, sizebuf, timebuf);
    }
}

```

```

    return S3StatusOK;
}

S3ListBucketHandler listBucketHandler =
{
    responseHandler,
    &listBucketCallback
};
S3_list_bucket(&bucketContext, NULL, NULL, NULL, 0, NULL, &listBucketHandler, NULL);

```

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!

```

S3_delete_bucket(S3ProtocolHTTP, S3UriStylePath, access_key, secret_key, host, sample_bucket,

```

CREATING AN OBJECT (FROM A FILE)

This creates a file hello.txt.

```

#include <sys/stat.h>
typedef struct put_object_callback_data
{
    FILE *infile;
    uint64_t contentLength;
} put_object_callback_data;

static int putObjectDataCallback(int bufferSize, char *buffer, void *callbackData)
{
    put_object_callback_data *data = (put_object_callback_data *) callbackData;

    int ret = 0;

    if (data->contentLength) {
        int toRead = ((data->contentLength > (unsigned) bufferSize) ? (unsigned) bufferSize :
            ret = fread(buffer, 1, toRead, data->infile);
    }
    data->contentLength -= ret;
    return ret;
}

put_object_callback_data data;
struct stat statbuf;
if (stat(sample_file, &statbuf) == -1) {
    fprintf(stderr, "\nERROR: Failed to stat file %s: ", sample_file);
    perror(0);
    exit(-1);
}

int contentLength = statbuf.st_size;
data.contentLength = contentLength;

if (!(data.infile = fopen(sample_file, "r"))) {
    fprintf(stderr, "\nERROR: Failed to open input file %s: ", sample_file);
    perror(0);
    exit(-1);
}

S3PutObjectHandler putObjectHandler =

```

```

{
    responseHandler,
    &putObjectDataCallback
};

S3_put_object(&bucketContext, sample_key, contentLength, NULL, NULL, &putObjectHandler, &data

```

DOWNLOAD AN OBJECT (TO A FILE)

This downloads a file and prints the contents.

```

static S3Status getObjectDataCallback(int bufferSize, const char *buffer, void *callbackData)
{
    FILE *outfile = (FILE *) callbackData;
    size_t wrote = fwrite(buffer, 1, bufferSize, outfile);
    return ((wrote < (size_t) bufferSize) ? S3StatusAbortedByCallback : S3StatusOK);
}

S3GetObjectHandler getObjectHandler =
{
    responseHandler,
    &getObjectDataCallback
};
FILE *outfile = stdout;
S3_get_object(&bucketContext, sample_key, NULL, 0, 0, NULL, &getObjectHandler, outfile);

```

DELETE AN OBJECT

This deletes an object.

```

S3ResponseHandler deleteResponseHandler =
{
    0,
    &responseCompleteCallback
};
S3_delete_object(&bucketContext, sample_key, 0, &deleteResponseHandler, 0);

```

CHANGE AN OBJECT'S ACL

This changes an object's ACL to grant full control to another user.

```

#include <string.h>
char ownerId[] = "owner";
char ownerDisplayName[] = "owner";
char granteeId[] = "grantee";
char granteeDisplayName[] = "grantee";

S3AclGrant grants[] = {
    {
        S3GranteeTypeCanonicalUser,
        {},
        S3PermissionFullControl
    },
    {
        S3GranteeTypeCanonicalUser,
        {},
        S3PermissionReadACP
    },
    {
        S3GranteeTypeAllUsers,
        {},
        S3PermissionRead
    }
};

```

```
strncpy(grants[0].grantee.canonicalUser.id, ownerId, S3_MAX_GRANTEE_USER_ID_SIZE);
strncpy(grants[0].grantee.canonicalUser.displayName, ownerDisplayName, S3_MAX_GRANTEE_DISPLAY_NAME_SIZE);

strncpy(grants[1].grantee.canonicalUser.id, granteeId, S3_MAX_GRANTEE_USER_ID_SIZE);
strncpy(grants[1].grantee.canonicalUser.displayName, granteeDisplayName, S3_MAX_GRANTEE_DISPLAY_NAME_SIZE);

S3_set_acl(&bucketContext, sample_key, ownerId, ownerDisplayName, 3, grants, 0, &responseHandler);
```

GENERATE OBJECT DOWNLOAD URL (SIGNED)

This generates a signed download URL that will be valid for 5 minutes.

```
#include <time.h>
char buffer[S3_MAX_AUTHENTICATED_QUERY_STRING_SIZE];
int64_t expires = time(NULL) + 60 * 5; // Current time + 5 minutes

S3_generate_authenticated_query_string(buffer, &bucketContext, sample_key, expires, NULL);
```