Object Drive 1.0
System Admin Guide

(SAG)

Refer to System Installation Procedures Guide for Configuration

Last Updated: February 25, 2019

Master Encryption Key

The master encryption key is configured during initial setup as described in the System Installation Procedures Guide (SIPG)

For convenience, here is the description of this environment variable

OD_ENCRYPT_MASTERKEY *since v1.0*	The secret master key used as part of the encryption key for all files stored in the system. If this value is changed, all file keys must be adjusted at the same time. If you don't set this, the service will shut down. Note that if a token.jar is installed onto the system, we can use the encrypted indicator format like `ENC{...}

Every instance of Object Drive within the cluster MUST be using the same key.

Backing up Encryption Key

The encryption key should be backed up to a place where it can be restored as necessary in the event that all instances of Object Drive server are terminated and deleted. This may be maintained by an ISSO or other individual in charge of managing secrets in escrow.

Changing the Encryption Key

Periodically, the master encryption key may need to be changed.

1. **Backup the database!** This action may be long running and is irreversible. You should always make a snapshot or backup of the database before applying a system wide change like this or else you run the risk of losing all of the data. Typical disaster recovery plans (DRP) also call for verifying the backup, but that is outside the scope of this procedure.

2. **Stop all running instances of Object Drive**. To prevent new objects being created, or changes from the service transpiring during the update, all services should be stopped.

3. **Using the credentials to access the database**, login to the database using the mysql client.

4. **Set database to read only mode.** This step should only be performed on instances running version 1.0.2 (June 30, 2017) or newer. The services will not attempt to make changes if the value of schemaversion stored in the dbstate table is different than the

internally expected version as of v1.0.2.

```
update dbstate set schemaversion =
concat(replace(schemaversion,'READONLY',''),'READONLY');
```

5. **Verify that the rotate_keys procedure exists.** The following SQL statement can be typed in to validate existence of the procedure. A result of 1 indicates the procedure exists.

```
select count(0) from information_schema.routines
where routine_schema = database() and routine_name = 'rotate_keys';
```

If it does not exist, then you may create it as follows

```
CREATE PROCEDURE rotate_keys(IN oldMasterPass VARCHAR(255), IN newMasterPass
VARCHAR(255), IN entropy VARCHAR(255))
BEGIN
declare exit handler for sqlexception
begin
 rollback;
end;
declare exit handler for sqlwarning
begin
 rollback;
end;
start transaction;
update object_permission p set encryptKey = unhex(
 bitwise256_xor(
   new_keydecrypt( newMasterPass, hex(p.permissionIV)),
   bitwise256_xor(
     new_keydecrypt(oldMasterPass, hex(p.permissionIV)),
     hex(p.encryptKey)
   )
 )
);
update object_permission p set permissionMAC = unhex(new_keymac(newMasterPass,
p.grantee, p.allowCreate, p.allowRead, p.allowUpdate, p.allowDelete, p.allowShare,
hex(encryptKey)));
commit;
END;
```

4. **Make note of the current encryption key**. For a typical installation, this value can be found in the file `/opt/services/object-drive-1.0/env.sh` named OD_ENCRYPT_MASTERKEY.

5. **Determine a new value to use as the master encryption key**. As a convenience, you can generate a hash from the command line, or a simple hex of the current time in MySQL as follows. Whatever approach you take to determine your new encryption key, make sure it follows guidelines of strong and secure passwords outlined by your organization. The ISSO should backup the new encryption key.

   ```
   select hex(current_timestamp());
   ```

   A sample run of the above produces output similar to the following:
   `323031382D30382D30372031383A33383A3133`

   If you want something longer, you can use the following which includes some additional database specific functions and trims the left and right to produce a desired length.

   ```
   select lcase(reverse(right(left(concat(hex(current_timestamp()), hex(database()),
   hex(session_user())), 80), 60)));
   ```

   A sample run of the above produces output similar to the following:
   `e2237313042756375726462646164716461647564756d65343a36333a3831302`

6. **Execute the change**. You can call the procedure as follows replacing the currentKey and newKey with the string values identified in steps 4 and 5 respectively:

   ```
   CALL rotate_keys('currentKeyFromStep4', 'newKeyFromStep5', 'reserved-for-entropy');
   ```

   This operation may take some time to complete as it will update all permissions on all objects stored in this system. The larger the instance, the more time it will take to complete.

   At present, there is no batch processing for this change, and all changes take place in a single transaction.

   There is no indication of success or failure until attempting to access a file. If you provide the wrong key in step 4, the data may be irrevocably lost. It is critical to ensure that you've taken a backup in step 1.

7. **Start service instances**. If you stopped the object drive server instances, you may restart them after updating the OD_ENCRYPT_MASTERKEY value in /opt/services/object-drive-1.0/env.sh with the new master key derived in step 5. If you

chose to put instances in read only mode, then you should shut down all instances now before making the change

8. **Perform a Test**. Now is the time to do a test to verify that you and/or other users can continue to access the streams of files you were able to before. If you are able to download a file and open it, then the change was successful.

   However, if a file you were able to access previously now downloads as complete gibberish, then something went wrong with the key rotation. If this happens, it's recommended to restore the OD_ENCRYPT_MASTERKEY to the value it was before, and restore the database snapshot and re-attempt. If you continue to have problems, reach out to developers.

9. **Enable Write Access**. Now you can set the write access to the database by changing the schema version back to what it was before. This step is only necessary if you set object drive to read only in step 4.

   ```
   update dbstate set schemaversion = replace(schemaversion,'READONLY','');
   ```

10. **Done**.

FIPS 140-2

From wikipedia

> The Federal Information Processing Standard (FIPS) Publication 140-2, (FIPS PUB 140-2), is a U.S. government computer security standard used to approve cryptographic modules. The title is Security Requirements for Cryptographic Modules. Initial publication was on May 25, 2001 and was last updated December 3, 2002

Object Drive is built using the Go programming language, freely available from Google at https://golang.org/dl/ or with BoringCrypto enabled versions, available here: https://go-boringcrypto.storage.googleapis.com/

Natively, the Go programming language does not support FIPS 140-2 compliance. Per the National Institute of Standards and Technology (NIST), compliance is through use of those modules that have been successfully vetted through the Cryptographic Module Validation Program (CMVP).

The Go programming language has branches, starting with Go version 1.8, that include alterations in the crypto and other packages to leverage the core of the BoringSSL codebase as a module named BoringCrypto. This module was validated through NIST and has Certificate #2964 for software version 24e5886c0edfc409c8083d10f9f1120111efd6f5. The branch of go is 'dev.boringcrypto'.

Verifying the Artifacts Used to Compile

The boringssl code for that software version may be downloaded from https://commondatastorage.googleapis.com/chromium-boringssl-docs/fips/boringssl-24e5886c0edfc409c8083d10f9f1120111efd6f5.tar.xz with a sha256 checksum of 15a65d676eeae27618e231183a1ce9804fc9c91bcc3abf5f6ca35216c02bf4da. This download and check is done within the Go programming language branches incorporating BoringCrypto as seen here: https://github.com/golang/go/blob/dev.boringcrypto.go1.11/src/crypto/internal/boring/build/build.sh . This compiles the module written in C, and CGO capabilities allows it to be called from Go. When a program is compiled using a version of Go that has BoringCrypto baked in, it makes callouts to the approved module in place of the native code to do the same.

Compare with MD5SUM from Google Bucket

The gsutil utility, available as part of the google-cloud-sdk available from google can be used to report on the MD5sum of a file as stored in the google bucket, as well as calculate the same locally. For example, running this command:

```
gsutil stat gs://go-boringcrypto/go1.11b4.linux-amd.tar.gz
```

reports the md5 hash to be `/yvx0VhhFQjMaeoc1ce/kg==`, which is the base64 of the 128bit hash. The hex equivalent is `ff2bf1d158611508cc69ea1cd5c7bf92` . This is the value you'd get if you downloaded the file using wget, and then ran md5sum on the file. You can also convert from base64 to hex like this

```
echo "/yvx0VhhFQjMaeoc1ce/kg==" | base64 -d | xxd -p
```

which will return the hex equivalent that md5sum will report.

```
ff2bf1d158611508cc69ea1cd5c7bf92
```

Compare with SHA256SUM

A sha256sum comparison can be done on the file as well, and the correct expected hash values are maintained on the RELEASES page for the dev.boringcrypto branch found here:
https://go.googlesource.com/go/+/dev.boringcrypto/misc/boring/RELEASES
This is an HTML based resource from the Google git repository. For text, add `?format=text` to the end, which will return it base64 encoded. To Get just the sha256sum part of a line from a particular release, the following one liner can be used:

```
curl -s https://go.googlesource.com/go/+/dev.boringcrypto/misc/boring/RELEASES?format=text |
base64 -d | grep go1.11b4.linux-amd64.tar.gz | awk '{print $5}'
```

For the file downloaded locally, running

```
sha256sum go1.11b4.linux-amd.tar.gz
```

It should produce the following as the result

```
d53417b2071af0104fbc15a957000bccdcb5bbc094df0401f67d51968f7f2e4e
```

Verifying Binary Built with Go is using BoringCrypto

Since compiling a go program creates a single binary to run on the architecture it is compiled for, it's useful to have ways to verify what actually makes up that binary.

If the Go programming language is available on the system, you can run the following command on a previously produced binary to see what symbols it has in use

```
go tool nm <binary-name>
```

For example, `go tool nm odrive`. This will produce a lengthy output of the symbols used. To narrow down to boring crypto, you can grep the output as follows

```
go tool nm odrive | grep "crypto/internal/boring._cgo"
```

This will reflect those functions that are running through the boring crypto module. Some expected output should appear similar to the following (truncated for brevity)

```
17f58f0 D crypto/internal/boring._cgo_925316647872_Cfunc_EVP_AEAD_CTX_open_wrapper
 17f58f8 D crypto/internal/boring._cgo_925316647872_Cfunc_EVP_AEAD_CTX_seal_wrapper
 17f5900 D crypto/internal/boring._cgo_925316647872_Cfunc__Cmalloc
 17f5908 D crypto/internal/boring._cgo_925316647872_Cfunc__goboringcrypto_AES_cbc_encrypt
 17f5910 D crypto/internal/boring._cgo_925316647872_Cfunc__goboringcrypto_AES_ctr128_encrypt
 17f5918 D crypto/internal/boring._cgo_925316647872_Cfunc__goboringcrypto_AES_decrypt
 17f5920 D crypto/internal/boring._cgo_925316647872_Cfunc__goboringcrypto_AES_encrypt
 17f5928 D crypto/internal/boring._cgo_925316647872_Cfunc__goboringcrypto_AES_set_decrypt_key
 17f5930 D crypto/internal/boring._cgo_925316647872_Cfunc__goboringcrypto_AES_set_encrypt_key
 17f5938 D crypto/internal/boring._cgo_925316647872_Cfunc__goboringcrypto_BN_bin2bn
 17f5940 D crypto/internal/boring._cgo_925316647872_Cfunc__goboringcrypto_BN_free

...//snipped//...
```

Object Drive Reporting

Starting with v1.0.18, Object Drive will now report in a few different ways whether it was built with a version of Go that has the BoringCrypto module in place for crypto related calls

Command Line Flag

A command line flag "isfips" is added. It can be called as follows:

```
odrive[1] isfips
```

This will indicate a FIPS 140-2 compliance check for the BoringCrypto module availability and exit.

[1] The name of the binary is by default 'odrive' within the development environment and docker images. For RPM deployed instances, the name of this binary is object-drive-1.0

```
FIPS 140-2 compliance check for BoringCrypto module

Built with Go runtime.version: go1.11b4
       base_go_version = go1.11
       boringcrypto_enabled = b
       boringcrypto_update_version = 4
The version of Go used to compile this binary uses BoringCrypto

If Go is available, you may also check whether the binary is using symbols from the module
  go tool nm odrive | grep "crypto/internal/boring._cgo"
```

Sample Output from calling: odrive isfips

Logging

When started, logging output will also check the runtime.Version and report if its built using a version of the Go programming language that uses BoringCrypto. Here is a sample log entry

```
2018-11-02T11:28:26.608740536-04:00   INFO   bring-crypto   {"update": "4", "runtime.Version":
"go1.11b4"}
```

API Documentation

The API documentation includes the version, build number, and build date at the top of each page. When the service is built with a version of Go programming language that uses BoringCrypto, it will also include the boring crypto update suffix to the version number.

- With BoringCrypto
  - **v1.0.19b4 -** The 'b4' indicates it was built with BoringCrypto update 4
- Without BoringCrypto
  - **v1.0.19 -** This indicates a version of Object Drive built without BoringCrypto

RPM Filename

Like the API documentation, if the service is built with a version of Go programming language using BoringCrypto, it will also include the boring update suffix in the RPM filename as part of the version.   The following are examples with and without BoringCrypto respectively

- With BoringCrypto
  - object-drive-1.0-1.0.19**b4**-2934.20190220.x86_64.rpm
- Without BoringCrypto
  - object-drive-1.0-1.0.19-beta.2934.20190220.x86_64.rpm

Docker Images

Docker images tagged for a version will have a suffix of "-bc" indicating they include binaries built with BoringCrypto.

- With BoringCrypto:
  - deciphernow/**odrive-bc**:latest
  - docker-dime.di2e.net/dime/object-drive-server:**1.0.19b4**
  - docker-dime.di2e.net/dime/object-drive-server:**1.0.19b4-release-candidate**
- Without BoringCrypto
  - deciphernow/**odrive**:latest
  - docker-dime.di2e.net/dime/object-drive-server:**1.0.19**
- No guarantee of whether BoringCrypto is included or not:
  - docker-dime.di2e.net/dime/object-drive-server:**latest**

Use of a docker image without specifying the tag will result in the latest tag being used. This should never be done for a production environment, and should be considered unstable with no guarantee of what version it actually represents. Only tags for specific versions are supported.

Useful SQL Queries

See when schema was created

```
select create_time from information_schema.tables
where table_schema = database() and table_name = 'dbstate';
```

Check for presence of a function or procedure by name

```
select created, last_altered, routine_name
from information_schema.routines
where routine_schema = database() and routine_name = 'aacflatten';
```

List all tables in database, and space used

```
SELECT table_name, table_rows, data_length, index_length,
round(((data_length + index_length) / 1024 / 1024),2) "Size in MB"
FROM information_schema.TABLES where table_schema = database();
```

Size of the database

```
SELECT table_schema "Data Base Name",
sum( data_length + index_length) / 1024 / 1024 "Data Base Size in MB"
FROM information_schema.TABLES GROUP BY table_schema ;
```

Count of Objects by Document Size Ranges

```
      select '0 - 5MB', count(0) from object
      where contentsize >= 0 and contentsize < (5*1024*1024)
union select '5MB - 10MB', count(0) from object
      where contentsize >= (5*1024*1024) and contentsize < (10*1024*1024)
union select '10MB - 25MB', count(0) from object
      where contentsize >= (10*1024*1024) and contentsize < (25*1024*1024)
union select '25MB - 100MB', count(0) from object
      where contentsize >= (25*1024*1024) and contentsize < (100*1024*1024)
union select 'Larger than 100MB', count(0) from object
      where contentsize >= (100*1024*1024);
```

Here's an alternative to the above which also breaks down by object type and content type

```
     select '0 - 5MB' streamsize, ot.name typename, o.contentType, count(0) qty from object o
      inner join object_type ot on o.typeid = ot.id
      where o.contentsize >= 0 and contentsize < (5*1024*1024) group by 1,2,3
union select '5MB - 10MB', ot.name, o.contentType, count(0) from object o
      inner join object_type ot on o.typeid = ot.id
      where o.contentsize >= (5*1024*1024) and contentsize < (10*1024*1024) group by 1,2,3
union select '10MB - 25MB', ot.name, o.contentType, count(0) from object o
      inner join object_type ot on o.typeid = ot.id
      where o.contentsize >= (10*1024*1024) and contentsize < (25*1024*1024) group by 1,2,3
union select '25MB - 100MB', ot.name, o.contentType, count(0) from object o
      inner join object_type ot on o.typeid = ot.id
      where o.contentsize >= (25*1024*1024) and contentsize < (100*1024*1024) group by 1,2,3
union select 'Larger than 100MB', ot.name, o.contentType, count(0) from object o
      inner join object_type ot on o.typeid = ot.id
      where o.contentsize >= (100*1024*1024) group by 1,2,3;
```

Fixing User Authorization Object Cache from Snippets

If users are not able to see folders and files shared to them due to an error that could have occurred during caching when running a version of Object Drive prior to 1.0.13

```
update useraocache set sha256hash = '0', iscaching = 0 where iscaching > 0 and
timestampdiff(minute, cachedate, now()) > 5;
```

You can also target a specific user as follows

```
update useraocache set sha256hash = '0', iscaching = 0 where userid in (
  select id from user where distinguishedname =
  'cn=test tester07,ou=people,ou=dae,ou=chimera,o=u.s. government,c=us'
);
```

Note that the issue that can cause this may happen again until an upgrade to the service is performed that addresses deadlocks at the database tier. After either command type is executed, when a user accesses ODrive again, it will start fresh with attempting to establish the cache of ACMs they can access based upon their user profile

Removing Duplicate User Cache Records

Under certain scenarios, it's possible that a parallel request won't be properly detected by the service and this can result in multiple user cache records being added. A subsequent restart of the service will then leave affected users unable to access the system as a key handler expects only one record to be present and will roll up a failure as a result.

These duplicate records may be periodically cleared by executing the following SQL statement

```
delete e.*
from useraocache e
where id in (
  select id
  from (
    select id
    from useraocache
    where userid in (
      select a.userid
      from (
        select userid, count(0)
        from useraocache
        group by userid
        having count(0) > 1
      ) as a
    ) and id not in (
      select maxid
      from (
        select ua.id, max(ua.id) maxid, hex(ua.userid)
        from useraocache ua inner join (
          select userid, count(0)
          from useraocache
          group by userid
          having count(0) > 1
        ) as a on ua.userid=a.userid
        group by ua.userid
      ) as b
    )
  ) as c
);
```

Brute Force Clearing User Cache and ACM Associations

To fully clear the cache AO for a user, and any existing associations to ACMs, the following queries can be performed

```
delete from useraocachepart
where userid = any(
  select id from user
  where distinguishedname = 'cn=test tester10,ou=people,ou=dae,ou=chimera,o=u.s. government,c=us'
);

delete from useraocache
where userid = any(
  select id from user
  where distinguishedname = 'cn=test tester10,ou=people,ou=dae,ou=chimera,o=u.s. government,c=us'
);

delete from useracm
where userid = any(
  select id from user
  where distinguishedname = 'cn=test tester10,ou=people,ou=dae,ou=chimera,o=u.s. government,c=us'
);
```

To apply this for all users, simply reduce the where clauses

```
delete from useraocachepart;

delete from useraocache;

delete from useracm;
```

It will take a few seconds to rebuild the associations for a user when they first access the system. Be forewarned that doing this may result in unforseen issues with concurrency as multiple users login/access Object Drive services in a small span of time.

Identifying and Fixing Faulty ACMS

An ACM is stored in multiple ways internal to Object Drive to improve search performance. A flaw in the way the data for ACM association was uncovered and known to affect 1.0.17, and 1.0.18, and likely several versions prior to that.

While 1.0.19b4 fixes the cause of this issue, it is prudent to check to see if there are any ACMs that may have been created in the system incorrectly.

To check for faulty ACMs run the following query

```
select id, flattenedacm from acm2 where id not in (select acmid from acmpart2);
```

If there are any rows returned, then the following script can be run to repair them. This will be included in a database migration as part of a future patch release.

```
DROP PROCEDURE IF EXISTS sp_Patch_20190225_fix_acmpart2_associations;
DELIMITER //
CREATE PROCEDURE sp_Patch_20190225_fix_acmpart2_associations()
BEGIN
    -- declare variables
    DECLARE vACMID int default 0;
    DECLARE vACMName text default '';
    DECLARE vSHA256Hash char(64) default '';
    DECLARE vKeyValuePart varchar(2000) default '';
    DECLARE vKeyID int default 0;
    DECLARE vKeyName varchar(255) default '';
    DECLARE vValueID int default 0;
    DECLARE vValueName varchar(255) default '';
    DECLARE vPartID int default 0;
    DECLARE c_acm2_finished int default 0;
    DECLARE c_acm2 cursor for SELECT id, flattenedacm
        from acm2 where id not in (select acmid from acmpart2);
    DECLARE continue handler for not found set c_acm2_finished = 1;

    INSERT INTO migration_status SET description = '20190225_fix_acmpart2_associations repairing acm2
rows that are missing acmpart2 rows';
    ACMPOPULATE: BEGIN

        -- get list of acm2 rows that dont have acmpart2 rows
        OPEN c_acm2;
        -- with each acm2 row needing fixed (loop)
        get_acm2: LOOP
            -- get row into variables
            FETCH c_acm2 INTO vACMID, vACMName;
            IF c_acm2_finished = 1 THEN
                CLOSE c_acm2;
                LEAVE get_acm2;
            END IF;
            INSERT INTO migration_status SET description = concat('20190225_fix_acmpart2_associations
analyzing acm with id ', vACMID);
            -- with each part (loop)
            get_acmnamepart2: LOOP
                -- check if still processing
                IF length(vACMName) = 0 THEN
                    LEAVE get_acmnamepart2;
                END IF;
                -- get next part as delineated by semicolon
                SET vKeyValuePart := substring_index(substring_index(vACMName, ';', 1), ';', -1);
                -- remove it from the head along with trailing semicolon leaving the tail to process on
next loop
                SET vACMName := substr(vACMName, length(vKeyValuePart)+2);
                -- get key name
                SET vKeyName := substring_index(substring_index(vKeyValuePart, '=', 1), '=', -1);
                -- remove key from the part with trailing equal sign leaving only the values
                SET vKeyValuePart := substr(vKeyValuePart, length(vKeyName)+2);
                -- insert the key name and get its id
                IF (SELECT 1=1 FROM acmkey2 WHERE name = vKeyName) IS NULL THEN
                    INSERT INTO migration_status SET description =
concat('20190225_fix_acmpart2_associations adding missing acm key ', vKeyName, ' to acmkey2');
                    INSERT INTO acmkey2 (name) VALUES (vKeyName);
                    SET vKeyID := LAST_INSERT_ID();
                ELSE
                    SELECT id INTO vKeyID FROM acmkey2 WHERE name = vKeyName LIMIT 1;
                END IF;
                -- with each key/value (loop)
                get_acmvalue2: LOOP
                    -- check if still processing values
                    IF length(vKeyValuePart) = 0 THEN
                        LEAVE get_acmvalue2;
                    END IF;
                    -- get next value as delineated by comma
                    SET vValueName := substring_index(substring_index(vKeyValuePart, ',', 1), ',', -1);
                    -- remove it from the head along with trailing comma leaving the tail to process on
next loop
                    SET vKeyValuePart := substr(vKeyValuePart, length(vValueName)+2);
                    -- insert the value name and get its id
                    IF (SELECT 1=1 FROM acmvalue2 WHERE name = vValueName) IS NULL THEN
                        INSERT INTO migration_status SET description =
concat('20190225_fix_acmpart2_associations adding missing acm value ', vValueName, ' to acmvalue2');
                        INSERT INTO acmvalue2 SET name = vValueName;
                        SET vValueID := LAST_INSERT_ID();
                    ELSE
                        SELECT id INTO vValueID FROM acmvalue2 WHERE name = vValueName LIMIT 1;
                    END IF;
                    -- create association row between acm, key and value
                    IF (SELECT 1=1 FROM acmpart2 WHERE acmid = vACMID and acmkeyid = vKeyID and
acmvalueid = vValueID) IS NULL THEN
                        INSERT INTO migration_status SET description =
concat('20190225_fix_acmpart2_associations for acm with id ', vACMID, ': associating key ',vKeyName,'
with value ',vValueName);
                        INSERT INTO acmpart2 (acmid, acmkeyid, acmvalueid) VALUES (vACMID, vKeyID,
vValueID);
                        SET vPartID := LAST_INSERT_ID();
                    ELSE
                        SELECT id INTO vPartID FROM acmpart2 WHERE acmid = vACMID and acmkeyid = vKeyID
and acmvalueid = vValueID LIMIT 1;
                    END IF;
                -- end loop processing values
                END LOOP get_acmvalue2;
            -- end loop for getting part
            END LOOP get_acmnamepart2;
        -- end loop for acm2
        END LOOP get_acm2;
    END ACMPOPULATE;
    INSERT INTO migration_status SET description = '20190225_fix_acmpart2_associations data repairing
acm2 rows that are missing acmpart2 rows is done.';
END;
//
DELIMITER ;
CALL sp_Patch_20190225_fix_acmpart2_associations();
DROP PROCEDURE IF EXISTS sp_Patch_20190225_fix_acmpart2_associations;
```