Transactions in JStore

# General Approach

All writes to LXPs are written to a shadow directory within buckets. This shadow directory is called *TRANSACTIONS*. The files that are written are normal LXP records each stored in a separate file and identified in the normal manner by their oid.

When transactions are started a transaction start record is written to a per-store bucket called *Transactions* stored in a repository called *Transaction*\_*repository*. This start record is stored in a normal LXP and contains a single key labelled “START” and the transaction\_id as the corresponding value. This is strictly not necessary but makes recovery following a JVM failure faster.

On commit a commit record is written to the *Transaction*\_*repository. Once this has been written the transaction is durable. The commit record contains a single key labelled “LOG” the corresponding value is a serialised set of:*

*repository name X bucket name X oid*

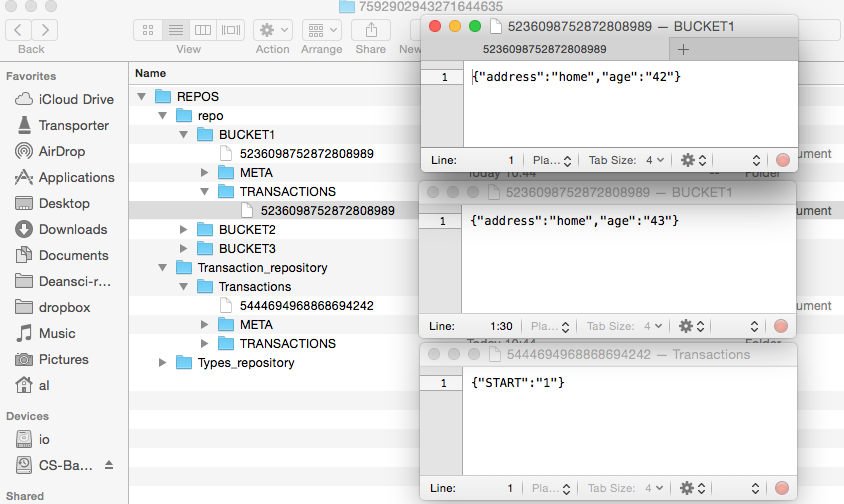
these may be used to find the shadow copies with which to commit the transaction should failure occur during the commit and after the record has been written. In order to be able to tell if the commit record has been written fully the LOG value contains start and end markers. Without these markers the commit record is invalid.

# Example Transaction

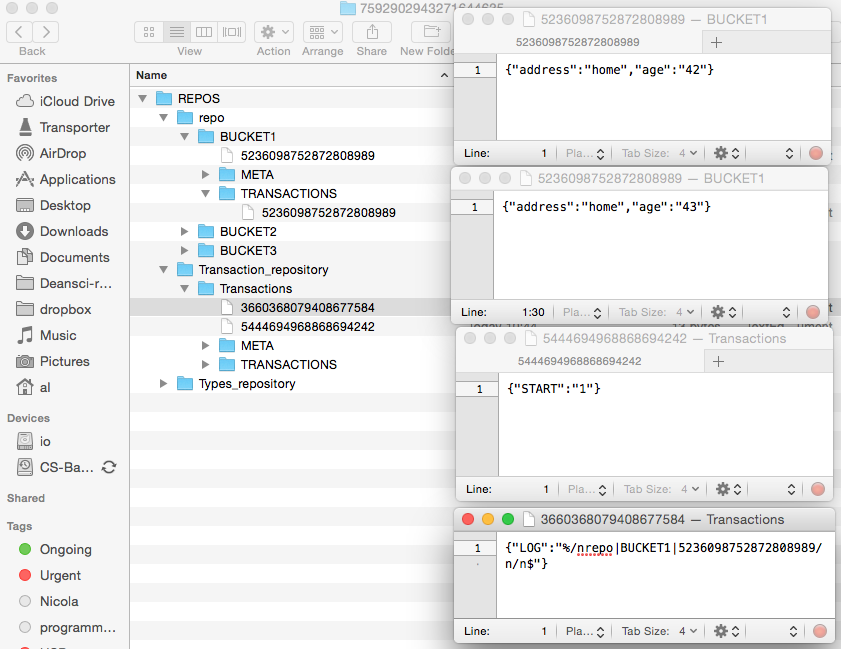
@Test  
**public synchronized void** testSimpleTransaction() **throws** RepositoryException, IllegalKeyException, BucketException, StoreException, TransactionFailedException {  
  
 IBucket b = **repo**.getBucket(*generic\_bucket\_name1*);  
  
 LXP lxp = **new** LXP();  
 lxp.put(**"age"**, **"42"**);  
 lxp.put(**"address"**, **"home"**);  
 b.makePersistent(lxp);  
 **long** oid = lxp.getId();  
  
 ITransaction txn = **store**.getTransactionManager().beginTransaction();  
  
 ILXP lxp2 = b.getObjectById(oid);  
 lxp2.put(**"age"**, **"43"**);  
 b.update(lxp2);

/\*\*\*\*\*\*\*\*\* A \*\*\*\*\*\*\*\*\*/  
 txn.commit();}

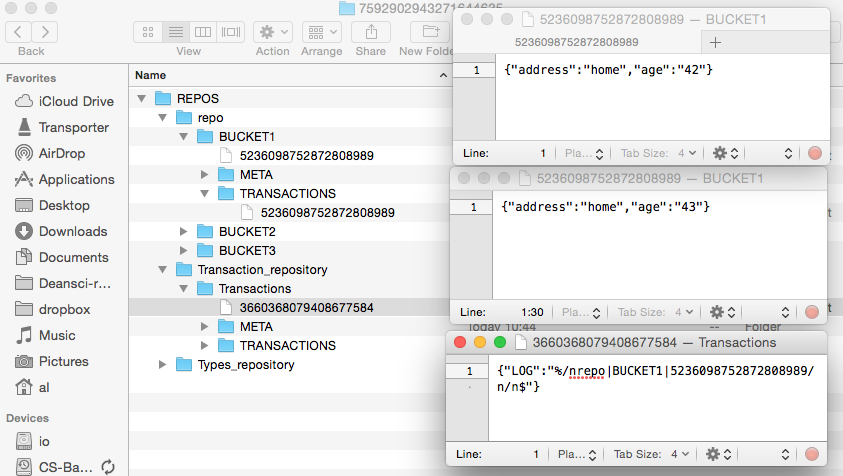
Structure of file system at point A in example above:



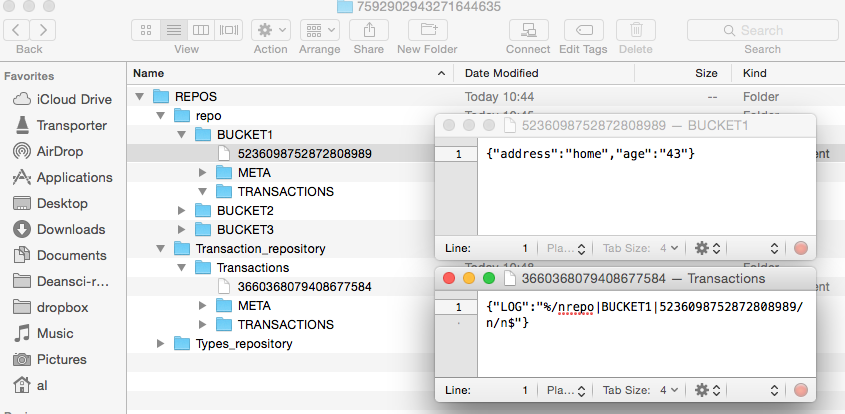
During Commit before completion (1):



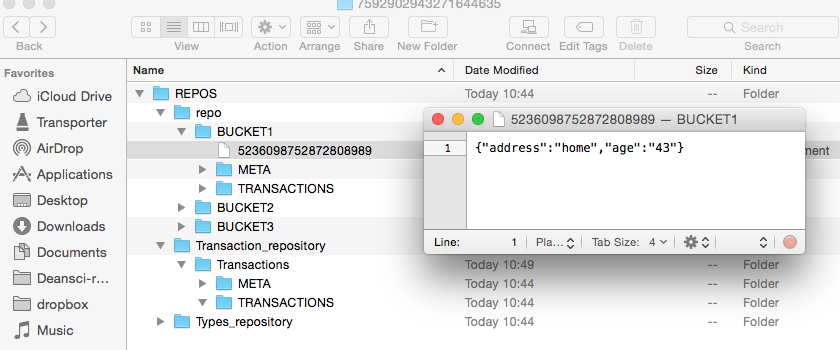
During commit before completion (2):



During commit before completion (3):



After commit has returned:



# JVM Failure recovery procedure

1. JVM crash before commit
   1. Leaves behind start records in the Transactions bucket in the Transaction\_repository repository
   2. Leaves files behind in TRANSACTIONS directory in buckets
2. JVM crash in commit before transaction log written.
   1. Leaves behind start records in the Transactions bucket in the Transaction\_repository repository
   2. Leaves files behind in TRANSACTIONS directory in buckets
3. JVM crash in commit after transaction log written – protects atomicity of transaction.
   1. May (or may not) Leaves behind start records in the Transactions bucket in the Transaction\_repository repository
   2. May leave files behind in TRANSACTIONS directory in buckets (can be found from the log)
   3. Leaves transaction log behind in Transaction repository

In all cases JVM recovery:

The need for cleanup is signalled by start records being present.

Cleanup should commit any transactions for which there is a complete transaction log in the Transactions bucket.

Some shadow copies may not be present in TRANSCATIONS bucket (shadow directory) if already installed – these can be safely ignored.

All other files are deleted.