DISASTER RECOVERY

Danger: The notes in this section are aimed at experts, making a best effort to recovery what they can from damaged filesystems. These steps have the potential to make things worse as well as better. If you are unsure, do not proceed.

JOURNAL EXPORT

Before attempting dangerous operations, make a copy of the journal like so:

```
cephfs-journal-tool journal export backup.bin
```

Note that this command may not always work if the journal is badly corrupted, in which case a RADOS-level copy should be made (http://tracker.ceph.com/issues/9902).

DENTRY RECOVERY FROM JOURNAL

If a journal is damaged or for any reason an MDS is incapable of replaying it, attempt to recover what file metadata we can like so:

```
cephfs-journal-tool event recover_dentries summary
```

This command by default acts on MDS rank 0, pass –rank=<n> to operate on other ranks.

This command will write any inodes/dentries recoverable from the journal into the backing store, if these inodes/dentries are higher-versioned than the previous contents of the backing store. If any regions of the journal are missing/damaged, they will be skipped.

Note that in addition to writing out dentries and inodes, this command will update the InoTables of each ‘in’ MDS rank, to indicate that any written inodes’ numbers are now in use. In simple cases, this will result in an entirely valid backing store state.

Warning: The resulting state of the backing store is not guaranteed to be self-consistent, and an online MDS scrub will be required afterwards. The journal contents will not be modified by this command, you should truncate the journal separately after recovering what you can.

JOURNAL TRUNCATION

If the journal is corrupt or MDSs cannot replay it for any reason, you can truncate it like so:

```
cephfs-journal-tool journal reset
```

Warning: Resetting the journal *will* lose metadata unless you have extracted it by other means such as `recover_dentries`. It is likely to leave some orphaned objects in the data pool. It may result in re-allocation of already-written inodes, such that permissions rules could be violated.

MDS TABLE WIPES

After the journal has been reset, it may no longer be consistent with respect to the contents of the MDS tables (InoTable, SessionMap, SnapServer).

To reset the SessionMap (erase all sessions), use:

```
cephfs-table-tool all reset session
```

This command acts on the tables of all ‘in’ MDS ranks. Replace ‘all’ with an MDS rank to operate on that rank only.

The session table is the table most likely to need resetting, but if you know you also need to reset the other tables then replace ‘session’ with ‘snap’ or ‘inode’.

MDS MAP RESET

Once the in-RADOS state of the filesystem (i.e. contents of the metadata pool) is somewhat recovered, it may be necessary to update the MDS map to reflect the contents of the metadata pool. Use the following command to reset the MDS map to a single MDS:

```
ceph fs reset <fs name> --yes-i-really-mean-it
```

Once this is run, any in-RADOS state for MDS ranks other than 0 will be ignored: as a result it is possible for this to result in data loss.

One might wonder what the difference is between ‘fs reset’ and ‘fs remove; fs new’. The key distinction is that doing a remove/new will leave rank 0 in ‘creating’ state, such that it would overwrite any existing root inode on disk and orphan any existing files. In contrast, the ‘reset’ command will leave rank 0 in ‘active’ state such that the next MDS daemon to claim the rank will go ahead and use the existing in-RADOS metadata.

RECOVERY FROM MISSING METADATA OBJECTS

Depending on what objects are missing or corrupt, you may need to run various commands to regenerate default versions of the objects.

```
# Session table
cephfs-table-tool 0 reset session
# SnapServer
cephfs-table-tool 0 reset snap
# InoTable
cephfs-table-tool 0 reset inode
# Journal
cephfs-journal-tool --rank=0 journal reset
# Root inodes ("/" and MDS directory)
cephfs-data-scan init
```

Finally, you can regenerate metadata objects for missing files and directories based on the contents of a data pool. This is a three-phase process. First, scanning *all* objects to calculate size and mtime metadata for inodes. Second, scanning the first object from every file to collect this metadata and inject it into the metadata pool. Third, checking inode linkages and fixing found errors.

```
cephfs-data-scan scan_extents <data pool>
cephfs-data-scan scan_inodes <data pool>
cephfs-data-scan scan_links
```

‘scan_extents’ and ‘scan_inodes’ commands may take a *very long* time if there are many files or very large files in the data pool.

To accelerate the process, run multiple instances of the tool.

Decide on a number of workers, and pass each worker a number within the range 0-(worker_m - 1).

The example below shows how to run 4 workers simultaneously:

```
# Worker 0
cephfs-data-scan scan_extents --worker_n 0 --worker_m 4 <data pool>
# Worker 1
cephfs-data-scan scan_extents --worker_n 1 --worker_m 4 <data pool>
# Worker 2
cephfs-data-scan scan_extents --worker_n 2 --worker_m 4 <data pool>
# Worker 3
cephfs-data-scan scan_extents --worker_n 3 --worker_m 4 <data pool>
```

```
# Worker 0
cephfs-data-scan scan_inodes --worker_n 0 --worker_m 4 <data pool>
# Worker 1
cephfs-data-scan scan_inodes --worker_n 1 --worker_m 4 <data pool>
# Worker 2
cephfs-data-scan scan_inodes --worker_n 2 --worker_m 4 <data pool>
# Worker 3
cephfs-data-scan scan_inodes --worker_n 3 --worker_m 4 <data pool>
```

It is **important** to ensure that all workers have completed the scan_extents phase before any workers enter the scan_inodes phase.

After completing the metadata recovery, you may want to run cleanup operation to delete ancillary data geneated during recovery.

```
cephfs-data-scan cleanup <data pool>
```

FINDING FILES AFFECTED BY LOST DATA PGS

Losing a data PG may affect many files. Files are split into many objects, so identifying which files are affected by loss of particular PGs requires a full scan over all object IDs that may exist within the size of a file. This type of scan may be useful for identifying which files require restoring from a backup.

Danger: This command does not repair any metadata, so when restoring files in this case you must *remove* the damaged file, and replace it in order to have a fresh inode. Do not overwrite damaged files in place.

If you know that objects have been lost from PGs, use the `pg_files` subcommand to scan for files that may have been damaged as a result:

```
cephfs-data-scan pg_files <path> <pg id> [<pg id>...]
```

For example, if you have lost data from PGs 1.4 and 4.5, and you would like to know which files under /home/bob might have been damaged:

```
cephfs-data-scan pg_files /home/bob 1.4 4.5
```

The output will be a list of paths to potentially damaged files, one per line.

Note that this command acts as a normal CephFS client to find all the files in the filesystem and read their layouts, so the MDS must be up and running.

USING AN ALTERNATE METADATA POOL FOR RECOVERY

Warning: There has not been extensive testing of this procedure. It should be undertaken with great care.

If an existing filesystem is damaged and inoperative, it is possible to create a fresh metadata pool and attempt to reconstruct the filesystem metadata into this new pool, leaving the old metadata in place. This could be used to make a safer attempt at recovery since the existing metadata pool would not be overwritten.

Caution: During this process, multiple metadata pools will contain data referring to the same data pool. Extreme caution must be exercised to avoid changing the data pool contents while this is the case. Once recovery is complete, the damaged metadata pool should be deleted.

To begin this process, first create the fresh metadata pool and initialize it with empty file system data structures:

```
ceph fs flag set enable_multiple true --yes-i-really-mean-it
ceph osd pool create recovery <pg-num> replicated <crush-rule-name>
ceph fs new recovery-fs recovery <data pool> --allow-dangerous-metadata-overlay
cephfs-data-scan init --force-init --filesystem recovery-fs --alternate-pool recovery
ceph fs reset recovery-fs --yes-i-really-mean-it
```

```
cephfs-table-tool recovery-fs:all reset session
cephfs-table-tool recovery-fs:all reset snap
cephfs-table-tool recovery-fs:all reset inode
```

Next, run the recovery toolset using the –alternate-pool argument to output results to the alternate pool:

```
cephfs-data-scan scan_extents --alternate-pool recovery --filesystem <original filesystem nam
cephfs-data-scan scan_inodes --alternate-pool recovery --filesystem <original filesystem name
cephfs-data-scan scan_links --filesystem recovery-fs
```

If the damaged filesystem contains dirty journal data, it may be recovered next with:

```
cephfs-journal-tool --rank=<original filesystem name>:0 event recover_dentries list --alterna
cephfs-journal-tool --rank recovery-fs:0 journal reset --force
```

After recovery, some recovered directories will have incorrect statistics. Ensure the parameters mds_verify_scatter and mds_debug_scatterstat are set to false (the default) to prevent the MDS from checking the statistics, then run a forward scrub to repair them. Ensure you have an MDS running and issue:

```
ceph daemon mds.a scrub_path / recursive repair
```