# Commit Protocols

As the threads in Databus can be modeled as either producer or consumer in some cases(both eg: ***MergedStreamService***) there are two kinds of commit protocols in -

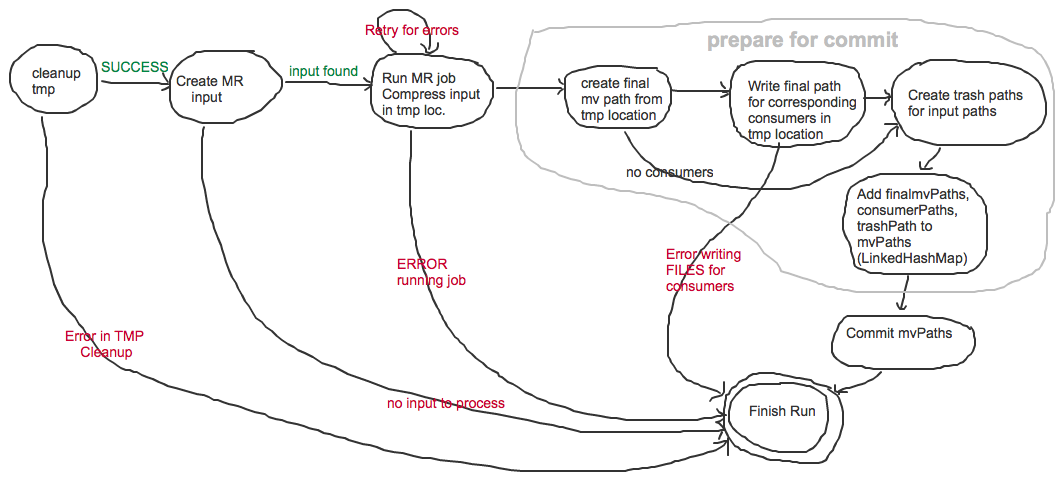
1. Producer Commit
2. Consumer Commit

# Threads in Databus and their types

1. **LocalStreamService** – Producer only thread. Takes data from HDFS and publishes localstreams, writes paths for consumers if any.
2. **MergedStreamService** – Both Consumer and Producer. Consumes the path’s written by localStreamService and produces paths for mirroredStreamService.
3. **MirrorStreamService** – Consumer only thread. Consumes path’s produced by mergedStreamService

## LocalStreamService State Machine

Following diagrams highlights the states through which LocalStreamService moves through to complete and publish data for a run.

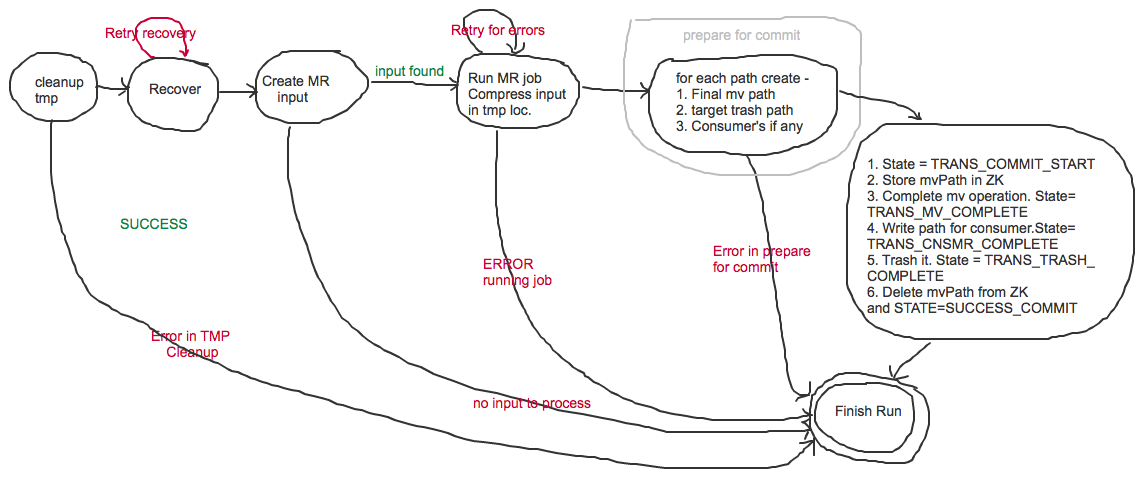


## Issues in Current Design

1. If databus has a catastrophic failure in **PREPARE for Commit phase** “***Write Final path to consumers”*** it leaves **Consumers**(*MirrorStreamService, MergedStreamService*) of LocalStreamService with paths that don’t exist. This can stall databus entirely from further processing. (*Consumer’s threads use DISTCP with –f option which fails if one path in the list in invalid*)
2. MergedStreamService running in another thread could get the consume file paths before LocalStreamService finishes **Commit mvPaths.** This can again result in Databus being Stalled.

## Proposed Design Changes in Producer Commit Protocol

The following diagram illustrates modified state machine for LocalStreamService. Note a state named Recovery which brings the producer to a consistent state if it failed in COMMIT in the last run.

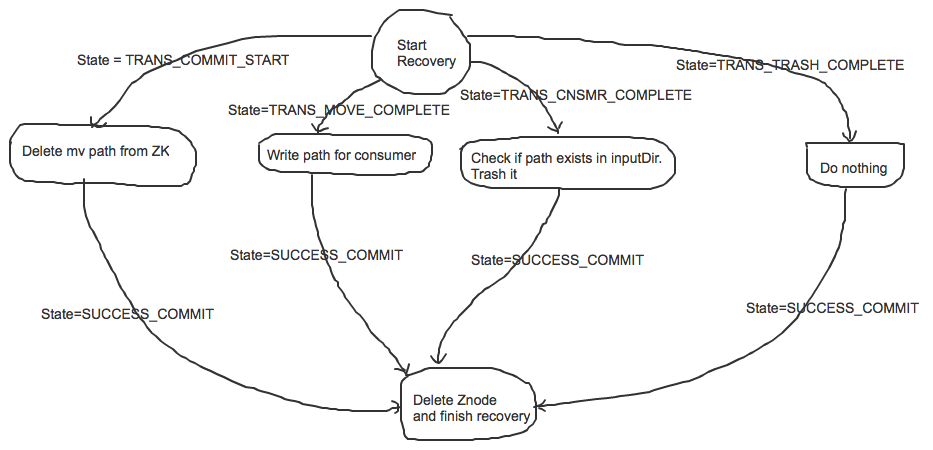


## New Producer Protocol Explained

1. Create a ZNODE in ZK with the thread name under /databus/<threadname>
2. Store state=TRANS\_COMMIT\_START in zNode
3. Store the Path object which contains(src, finalPath, trashPath, consumers for this path) in ZK
4. Move the path from temporaryLocation to DATABUS publish location - /databus/streams\_local/
5. Update state = TRANS\_MV\_COMPLETE in ZK
6. Write Path for it’s consumer’s
7. Update State = TRANS\_CNSMR\_COMPLETE
8. Move Path to it’s trash Location
9. Update State = TRANS\_TRASH\_COMPLETE
10. Delete the path object from ZNODE of this thread and update STATE=SUCCESS\_COMMIT

## Recovery for Producer Commit Protocol

The following diagram illustrates the state machine for Recovery mode for a Prodcuer Commit Protocol



1. if state=**TRANS\_COMMIT\_START** – Nothing was done delete the Path from ZK and exit recovery
2. if state=**TRANS\_MOVE\_COMPLETE** – Write Path for consumer’s and exit Recovery. Since writing to a file and updating ZK are individual distributed operations it is not possible to be sure whether this path was written to a consumer. This can cause a replay for single PATH.
3. If state=**TRANS\_CNSMR\_COMPLETE** – Move the path from inputDir i.e. /databus/data/<category>/<collector>/…. To TRASH location and exit recovery
4. If state=**TRANS\_TRASH\_COMPLETE** – Exit Recovery
5. If there is no ZNODE for a thread or it doesn’t contain any state then NO\_RECOVERY required.

## What is solved in Producer Commit Protocol

1. Databus will never stall as we write consumer paths after publishing them in localstreams
2. Consumer threads getting the consume path file prior to it getting published will never occur.

## Can data replay happen

1. In the above recovery protocol if we are in if state=**TRANS\_MOVE\_COMPLETE** then we are not sure whether databus died before writing CONSUMER PATHS or after writing CONSUMER PATHS and before updating state= **TRANS\_CNSMR\_COMPLETE.**
2. Databus will write the path to consumer path to break the ambiguity to avoid DATA LOSS.