KIP-696: Update Streams FSM to clarify ERROR state meaning

Status

Current state: *Accepted*

Discussion thread: here

JIRA: *here, here*

Please keep the discussion on the mailing list rather than commenting on the wiki (wiki discussions get unwieldy fast).

Motivation

Currently the ERROR state in KStreams means that there are no threads running. However, in KIP-663 streamThreads can be added dynamically so this definition no longer is useful. In that kip, the automatic transition to ERROR state upon the death of all threads is removed. Also, ERROR state should be terminal. This KIP should clarify the use of ERROR state as well as bring it into alignment with the other states

Public Interfaces

KafkaStreams will have added State PENDING_ERROR.

The following transitions will be added:

- RUNNING PENDING_ERROR
- REBALANCING PENDING_ERROR
- PENDING_ERROR ERROR

The following transitions will be removed:

- ERROR PENDING_SHUTDOWN

The SHUTDOWN_CLIENT option in the Streams Uncaught Exception Handler should leave the client state in ERROR instead of NOT_RUNNING

Proposed Changes

As ERROR will now be a terminal state, PENDING_ERROR will be added. The only way to reach the ERROR state is through the PENDING_ERROR state. This will mirror PENDING_SHUTDOWN and mean that resources are closing before the client transitions to ERROR. Currently, the client goes to ERROR before it closes the resources and does not signal when done.

ERROR will be redefined to mean that the streams client is in an unrecoverable state and should not be restarted until the problem has been investigated. Streams will only reach ERROR state in the event of an exceptional failure in which the `StreamsUncaughtExceptionHandler` chose to either shutdown the application or the client. After beginning to shutdown either the client or the application, the state of the streams client will be PENDING_ERROR until it has finished closing all resources. After doing so it will transition to ERROR. It is not recommended to automatically restart from ERROR state.

In order to be consistent, SHUTDOWN_CLIENT will leave the client state in ERROR instead of NOT_RUNNING. ERROR should be the state that exceptional failures leave the application in, not NOT_RUNNING.

Close() called on ERROR or PENDING_ERROR will be no-op and not throw an exception but we will log a warning.

Compatibility, Deprecation, and Migration Plan

- *These changes will affect the state machine. However this has already changed this release.*
- *The old behavior is already deprecated*
- *A compatibility issue: today users can call close() in the global state transition listener and it's common to try shutting down the application automatically when in ERROR (assuming it was not a terminating state). To solve this we will make the changes to close() mentioned above.*

Rejected Alternatives

- not having PENDING_ERROR