KIP-462: Use local thread id for KStreams

Status

Current state: *Accepted (2.3.0)*

Discussion thread: TBD

JIRA: Unable to render Jira issues macro, execution error.

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

Motivation

As we are rolling out static membership for general Kafka client applications, we need to have persistent `group.instance.id` throughout restarts. One of the edge case is that we are currently using shared static integer field for thread-id incrementation for multiple stream instances on one single JVM. So it is very easily to fall into this example scenario with two stream instance A, B:

Instance A init one stream thread Instance A init one stream thread Instance B init one stream thread Instance B init one stream thread

So instance ids being generated when two instances starts are : A-1, A-2, B-3, B-4

During one bounce, the order of initialization could change to:

Instance A init one stream thread Instance B init one stream thread Instance A init one stream thread Instance B init one stream thread

And then change instance ids to : A-1, A-3, B-2, B-4

This means we are creating 4 new instance ids compared with previous round, so broker could no longer leverage the static information it stored. Basically, the scenario invalidates static membership effectiveness.

*Another example would be: if a user `closes()` a `KafkaStreams` client and creates a new one (for example to recover failed threads), while the JVM is still running, it is more intuitive that the thread names are number from 1 to X again, and not from X+1 to 2*x on restart.*

Proposed Changes

Each stream instance will use local counter to create stream thread name. So the above example will be always generating instance ids as: *A-1, A-2, B-1, B-2. By persisting instance ids generation, the static membership will still behave as expected when we configure > 1 stream instances in one JVM.*

The original goal for using static integer to increment thread.id is to workaround the case where two stream instances configure same client.id in one JVM, in order to avoid conflict thread names. And note that, semantically we would not forbid users to set the same client.ids for throttling purposes for example. After this KIP, since this guard of fencing same client.ids of instances is removed, we will augment the client.id config description by stating what users should expect client.id to be propagated to internal embedded clients, and therefore what's the expected outcome if they choose to set same client.ids for different Streams client.

Public Interfaces

This change is internal, however it will affect KStream external information exposure. See next section.

Compatibility, Deprecation, and Migration Plan

- There is no anticipated negative impact. The thread-id is used to construct many stream component identifiers such as producer/consumer /stream thread names. If user exposes these information in their production environment, upgraded service may see different composed ids from previous round. Note that this only applies to multi-instance one JVM scenario and only one time, thus not going to globally affect things too much in general. The only pattern for this use case seems to be dynamic scaling, and we should actually void this pattern by adding a `stopThread()` and `addThread()` method to `KafkaStreams` directly.
- No deprecation/migration needed for this change.

Rejected Alternatives

N/A