KIP-292: Add transformValues() method to KTable

- Status
- Motivation
- Public Interfaces
- Proposed Changes
- Compatibility, Deprecation, and Migration Plan
- Test Plan
- Rejected Alternatives

Status

Current state: Accepted

Discussion thread: here

JIRA: KAFKA-6849

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

Motivation

In Kafka Stream the KStream interface has several transformValues methods, where as the KTable interface does not. The related mapValues method is available on both KTable and KStream. The transformValue method is more flexible than mapValues:

- Stateful implementations. transformValues accepts a supplier of transformers, rather than the mapper instance mapValues accepts. The supplier is used to instantiate a transformer per stream task, meaning the implementation does *not* need to be thread-safe. Conversely, mapValue shares the passed in mapper instance across stream tasks, so the implementations must be thread-safe.
- State-store access: transformValues allows access to existing state-stores, where as mapValues does not.
- Richer API: the ValueTransformer interface is much richer than the ValueMapper, supporting both init() and close() calls.

There is no conceptual reason not to support the richer transformValues on the KTable interface.

Public Interfaces

The following methods would be added to the Java KTable interface:

```
New KTable methods

<pr
```

The following methods would be added to the Scala KTable class.

Additional Scala KTable methods

Proposed Changes

The new methods on `KTableImpl` will add a new KTableTransformValues processor node and attach any state stores. The new KTableTransformV alues will be implemented in a similar manner to other processors, instantiating the user supplied transformer once per task.

The `ProcessorContext` passed to the `init()` method of transformer implementations will be restricted: any call to any variant of the `forward()` method will throw a `StreamsException`. This will stop implementations outputting values with a new key.

Compatibility, Deprecation, and Migration Plan

Users must upgrade to new version if they want to use this functionality.

Test Plan

Unit tests to cover new classes and methods. Integration or system test are not required.

Rejected Alternatives

Include overloads that take a `ValueTransformerSupplier`, matching the overloads available on `KStream`. These were no included as it keeps the interface more succinct, users can ignore the key value if they do not need it, and likely these overloads on the `KStream` interface will be deprecated, in favour of the 'WithKey' variants, in time.