# KIP-313: Add KStream.flatTransform and KStream.flatTransformValues

## Status

**Current state**: *Accepted*

**Discussion thread**: here

**JIRA**:

- ⚠ Unable to render Jira issues macro, execution error.
- ⚠ 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

Currently, `KStream.transform` transforms each record of the input stream into zero or more records in the output stream. A `Transformer` object is applied on each input record. The transformation is stateful, i.e., the transformer accesses state stores to compute output records. The signature of `KStream.transform` specifies a single key-value pair as output record. However, the user can emit multiple records for each input record if she calls for each desired output record method `forward` on the `ProcessorContext` object that is given to the transformer. By calling `forward` on the context, the user looses strong typing of the output records, i.e., if a user emits an output record that is not of the type specified in the transformer, the error is only thrown at run time and not at compile time. Furthermore, the public API misses a stateful transformation that transforms the value of each record of the input stream into zero or more records with same key but potentially different values in the output stream. Such a transformation would be the stateful equivalent of `KStream.flatMapValues`.

Stateful transform operations that ensure strong typing and that are more consistent with the stateless map operations, would make the public API safer to use, better comprehensible, and more complete.

## Public Interfaces

In org.apache.kafka.streams.kstream.KStream.java the following methods shall be added:

```
<K1, V1> KStream<K1, V1> flatTransform(
  final TransformerSupplier<? super K, ? super V, ? extends Iterable<? extends KeyValue<? extends K1, ? extends
V1>>> transformerSupplier,
  final String... stateStoreNames);

<VR> KStream<K, VR> flatTransformValues(final ValueTransformerSupplier<? super V, ? extends Iterable<? extends
VR>> valueTransformerSupplier,
                                        final String... stateStoreNames);

<VR> KStream<K, VR> flatTransformValues(
  final ValueTransformerWithKeySupplier<? super K, ? super V, ? extends Iterable<? extends VR>>
valueTransformerSupplier,
  final String... stateStoreNames);
```

## Proposed Changes

This KIP proposes the following changes:

- Addition of method `KStream.flatTransform` to the public API:
  The method transforms each record of the input stream into zero or more records in the output stream. It ensures strong typing by specifying in its signature a list of key-value pairs (i.e. `Iterable`) as output records for each input record. The addition of the method comprises the implementation of the method in the streams internals and appropriate tests.
- Addition of method `KStream.flatTransformValues` to the public API:
  The method transforms the value of each record of the input stream into zero or more records with same key but possible different values (with a possible new type) in the output stream and ensures strong typing. The addition of the method comprises the implementation of the method in the streams internals and appropriate tests.
- Restriction of method `KStream.transform` to emit zero or one output record:
  The restriction will merely be an adaptation of the API documentation with the recommendation to use `KStream.flatTransform` if multiple records should be emitted per input record. Disallowing the use of `ProcessorContext.forward` in the code would potentially break existing applications. However, for the next major release, it should be considered to disallow the use of `ProcessorContext.forward`. A JIRA issue that requires to disallow the use of `ProcessorContext.forward` in method `KStream.transform` will be created for the next major release.

## Compatibility, Deprecation, and Migration Plan

This change is compatible with existing Kafka Streams applications.

## Rejected Alternatives

- Disallow use of `ProcessorContext.forward` in the code:
  Disallowing the use of `ProcessorContext.forward` in the code would break existing applications. However, it should be considered for the next major release.