

# KIP-456: Helper classes to make it simpler to write test logic with TopologyTestDriver

- [Status](#)
- [Motivation](#)
- [Public Interfaces](#)
- [Proposed Changes](#)
- [Compatibility, Deprecation, and Migration Plan](#)
- [Rejected Alternatives](#)

## Status

**Current state:** Discarded in favor of [KIP-470: TopologyTestDriver test input and output usability improvements](#)

**Discussion thread:** [here](#)

JIRA:

A screenshot of a Jira error message. It features a yellow warning triangle icon on the left, followed by the text "Unable to render Jira issues macro, execution error." in a standard sans-serif font.

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

## Motivation

When using TopologyTestDriver you need to call ConsumerRecordFactory to create ConsumerRecord passed into pipeInput method to write to topic. Also when calling readOutput to consume from topic, you need to provide correct Deserializers each time.

You easily end up writing helper methods in your test classes, but this can be avoided when adding generic input and output topic classes wrapping existing functionality.

## Public Interfaces

## TestInputTopic

```
package org.apache.kafka.streams.test;

public class TestInputTopic<K, V> {
    //Timestamp handling
    //Record timestamp can be provided when piping input or use internally tracked time initialized with
constructors:
    //startTimestampMs the initial timestamp for generated records, if not provided uses current system time as
start timestamp.
    //autoAdvanceMs the time increment per generated record, if not provided auto-advance is disabled.

    //Constructors with serializers
    TestInputTopic(final TopologyTestDriver driver, final String topicName, final Serializer<K> keySerializer,
final Serializer<V> valueSerializer);
    TestInputTopic(final TopologyTestDriver driver, final String topicName, final Serializer<K> keySerializer,
final Serializer<V> valueSerializer, final long startTimestampMs);
    TestInputTopic(final TopologyTestDriver driver, final String topicName, final Serializer<K> keySerializer,
final Serializer<V> valueSerializer, final long startTimestampMs, final long autoAdvanceMs);
    //Constructors with serdes
    TestInputTopic(final TopologyTestDriver driver, final String topicName, final Serde<K> keySerde, final
Serde<V> valueSerde);
    TestInputTopic(final TopologyTestDriver driver, final String topicName, final Serde<K> keySerde, final
Serde<V> valueSerde, final long startTimestampMs);
    TestInputTopic(final TopologyTestDriver driver, final String topicName, final Serde<K> keySerde, final
Serde<V> valueSerde, final long startTimestampMs, final long autoAdvanceMs);

    //Advances the internally tracked time.
    void advanceTimeMs(final long advanceMs);

    //Methods to pipe single record
    void pipeInput(final V value);
    void pipeInput(final K key, final V value);
    void pipeInput(final V value, final long timestampMs);
    void pipeInput(final K key, final V value, final long timestampMs);
    void pipeInput(final K key, final V value, final Headers headers);
    void pipeInput(final K key, final V value, final Headers headers, final long timestampMs);

    //Methods to pipe list of records
    void pipeKeyValueList(final List<KeyValue<K, V>> keyValues);
    void pipeValueList(final List<V> values);
    void pipeKeyValueList(final List<KeyValue<K, V>> keyValues, final long startTimestamp, final long
advanceMs);
    void pipeValueList(final List<V> values, final long startTimestamp, final long advanceMs);
}
```

## TestOutputTopic

```
package org.apache.kafka.streams.test;

public class TestOutputTopic<K, V> {
    //Constructor with serializers
    TestOutputTopic(final TopologyTestDriver driver, final String topic, final Serde<K> keySerde, final
Serde<V> valueSerde);
    //Constructor with serdes
    TestOutputTopic(final TopologyTestDriver driver, final String topic, final Deserializer<K> keyDeserializer,
final Deserializer<V> valueDeserializer);

    //Methods to readOutput
    ProducerRecord<K, V> readRecord();
    KeyValue<K, V> readKeyValue();
    V readValue();

    //Output as collection
    Map<K, V> readKeyValuesToMap();
    List<KeyValue<K, V>> readKeyValuesToList();
    List<V> readValuesToList();
}
```

## Proposed Changes

This improvement adds TestInputTopic class which wraps TopologyTestDriver and ConsumerRecordFactory methods as one class to be used to write to Input Topics and TestOutputTopic class which collects TopologyTestDriver reading methods and provide typesafe read methods.

### Example

```
public class SimpleTopicTest {

    private TopologyTestDriver testDriver;
    private TestInputTopic<String, String> inputTopic;
    private TestOutputTopic<String, String> outputTopic;

    @Before
    public void setup() {
        testDriver = new TopologyTestDriver(TestStream.getTopology(), TestStream.getConfig());
        inputTopic = new TestInputTopic<>(testDriver, TestStream.INPUT_TOPIC, new Serdes.StringSerde(), new Serdes.
StringSerde());
        outputTopic = new TestOutputTopic<>(testDriver, TestStream.OUTPUT_TOPIC, new Serdes.StringSerde(), new
Serdes.StringSerde());
    }

    @After
    public void tearDown() {
        testDriver.close();
    }

    @Test
    public void testOneWord() {
        //Feed word "Hello" to inputTopic and no kafka key, timestamp is irrelevant in this case
        inputTopic.pipeInput("Hello");
        assertEquals(outputTopic.readValue(), "Hello");
        //No more output in topic
        assertNull(outputTopic.readRecord());
    }
}
```

- Examples in Testing Kafka Streams <https://kafka.apache.org/22/documentation/streams/developer-guide/testing.html> updated to use TestInputTopic and TestOutputTopic

## Compatibility, Deprecation, and Migration Plan

This is only adding new classes. The tests utilizing directly TopologyTestDriver can still be used. There are no compatibility issues.

## Rejected Alternatives

- It was considered to add methods to to return Iterable like in [KIP-451: Make TopologyTestDriver output iterable](#) , but it seems to be redundant with these List based methods