KIP-265: Make Windowed Serde to public APIs

Status

Current state: *Accepted*

Discussion thread: [DISCUSS] KIP-265: Make Windowed Serde to public APIs

JIRA:

key	summary	type	created	updated	due	assignee	reporter	priority	status	resolution

JQL and issue key arguments for this macro require at least one Jira application link to be configured

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

Motivation

Today windowed serdes are kept in internal packages currently, and hence for anyone trying to consume the topics writing by a windowed table, they need to implement their own serde implementations in order to read windowed keys.

In addition, for windowed serde they require inner serdes to deserialize the embedded data type, which is not supported right now.

To help on the user experience, it's better extracting the internal windowed serdes into the public package, and at the same time enable console consumer to set the window deserializers with the inner class.

Public Interfaces

This KIP would propose to add the following new config names into `StreamsConfig` and mention in the default serde configs that users should set them for window serdes:

```
/**
 * {@code default.windowed.key.serde.inner}
 */
public static final String DEFAULT_WINDOWED_KEY_SERDE_INNER_CLASS = "default.windowed.key.serde.inner";

/**
 * {@code default.windowed.value.serde.inner}
 */
public static final String DEFAULT_WINDOWED_VALUE_SERDE_INNER_CLASS = "default.windowed.value.serde.inner";

/** {@code default key.serde} */
/** {@code default value.serde} */
...
        + "Note when windowed serde class is used, one needs to set the inner serde class that implements the
<code>org.apache.kafka.common.serialization.Serde</code> interface via '"
        + DEFAULT_WINDOWED_KEY_SERDE_INNER_CLASS + "' or '" + DEFAULT_WINDOWED_VALUE_SERDE_INNER_CLASS + "' as
well";
```

And add the following class into the *o.a.k.streams.kstream* package :

```
public class WindowedSerdes {

    static public class TimeWindowedSerde<T> extends Serdes.WrapperSerde<Windowed<T>>;

    static public class SessionWindowedSerde<T> extends Serdes.WrapperSerde<Windowed<T>>;

    /**
     * Construct a {@code TimeWindowedSerde} object for the specified inner class type.
     */
    static public <T> Serde<Windowed<T>> timeWindowedSerdeFrom(final Class<T> type);

    /**
     * Construct a {@code SessionWindowedSerde} object for the specified inner class type.
     */
    static public <T> Serde<Windowed<T>> sessionWindowedSerdeFrom(final Class<T> type);
}


/**
 *  The inner serde class can be specified by setting the property
 *  {@link StreamsConfig#DEFAULT_WINDOWED_KEY_SERDE_INNER_CLASS} or
 *  {@link StreamsConfig#DEFAULT_WINDOWED_VALUE_SERDE_INNER_CLASS}
 *  if the no-arg constructor is called and hence it is not passed during initialization.
 */
public class TimeWindowedSerializer<T> implements Serializer<Windowed<T>>;

public class TimeWindowedDeserializer<T> implements Deserializer<Windowed<T>>;


public class SessionWindowedSerializer<T> implements Serializer<Windowed<T>>;

public class SessionWindowedDeserializer<T> implements Deserializer<Windowed<T>>;
```

And in `ConsoleConsumer` class, we let the deserializers to configure themselves, so that users can pass the inner serde as part of the message formatter properties with `key.deserializer.` and `value.deserializer.` prefix, which can then eventually be used by the deserailizer configuration logic to set the inner serde class.

```
val messageFormatterArgOpt = parser.accepts("property",
  "The properties to initialize the message formatter. Default properties include:\n" +
    "\tprint.timestamp=true|false\n" +
    "\tprint.key=true|false\n" +
    "\tprint.value=true|false\n" +
    "\tkey.separator=<key.separator>\n" +
    "\tline.separator=<line.separator>\n" +
    "\tkey.deserializer=<key.deserializer>\n" +
    "\tvalue.deserializer=<value.deserializer>\n" +
    "\nUsers can also pass in customized properties for their formatter; more specifically, users " +
    "can pass in properties keyed with \'key.deserializer.\' and \'value.deserializer.\' prefixes to configure
their deserializers.")
```

Note that for all windowed serdes, the inner serde can be either passed in during construction, or be configured in the `configure` call via the added inner serde classes.

Proposed Changes

As above, no inernal implementations would be affected.

Compatibility, Deprecation, and Migration Plan

This KIP only add new APIs and configs and hence should not affect any existing users.

Rejected Alternatives

None