KIP-393: Time windowed serde to properly deserialize changelog input topic

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

Status

Current state: *Accepted*

Discussion thread: Not available

JIRA: KAFKA-7110

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

Motivation

Currently the TimeWindowedSerde does not deserialize the windowed keys from a changelog topic properly. There is inconsistency on how the ChangeLoggingWindowBytesStore.java serializes the changelog windowed key and how the TimeWindowedSerde deserializes this key. The serde calls *from* in WindowKeySchema.java, which doesn't decode the serialized changelog windowed key properly since it doesn't take the sequence number into account.

1) In the from method of WindowKeySchema (called in deserialize in TimeWindowedDeserializer), we extract the window from the binary key, but we call getLong(binaryKey.length -TIMESTAMP_SIZE). However, ChangeLoggingWindowBytesStore will log the windowed key as:

```
changeLogger.logChange(WindowKeySchema.toStoreKeyBinary(key, timestamp, maybeUpdateSeqnumForDups()), value);
```

In toStoreKeyBinary, we store the key in

```
final ByteBuffer buf = ByteBuffer.allocate(serializedKey.length + TIMESTAMP_SIZE + SEQNUM_SIZE);
```

with the sequence number (used for de-duping). So the eventual result is that when we deserialize, we do not assume the windowed changelog key has a sequence number, and the window extracted will be gibberish since the bytes won't be alligned.

2) In the constructor of TimeWindowedDeserializer, the windowSize is fixed to Long.MAX_VALUE:

```
// TODO: fix this part as last bits of KAFKA-4468 public TimeWindowedDeserializer(final Deserializer<T> inner)
{ this(inner, Long.MAX_VALUE); }
public TimeWindowedDeserializer(final Deserializer<T> inner, final long windowSize) { this.inner = inner; this.
windowSize = windowSize; }
```

This will cause the end times to be improperly deserialized since the windowSize is used for constructing the TimeWindow:

```
public static <K> Windowed<K> from(final byte[] binaryKey, final long windowSize, final Deserializer<K>
deserializer, final String topic) {
    final byte[] bytes = new byte[binaryKey.length - TIMESTAMP_SIZE];
    System.arraycopy(binaryKey, 0, bytes, 0, bytes.length);
    final K key = deserializer.deserialize(topic, bytes); final Window window = extractWindow(binaryKey,
windowSize);
    return new Windowed<>(key, window);
}

private static Window extractWindow(final byte[] binaryKey, final long windowSize) {
    final ByteBuffer buffer = ByteBuffer.wrap(binaryKey);
    final long start = buffer.getLong(binaryKey.length - TIMESTAMP_SIZE);
    return timeWindowForSize(start, windowSize);
}

/**
 * Safely construct a time window of the given size,
 * taking care of bounding endMs to Long.MAX_VALUE if necessary
 */
public static TimeWindow timeWindowForSize(final long startMs,
                                           final long windowSize) {
    final long endMs = startMs + windowSize;
    return new TimeWindow(startMs, endMs < 0 ? Long.MAX_VALUE : endMs);
}
```

Proposed Changes

The current TimeWindowedSerde constructor:

```
public TimeWindowedSerde(final Serde<T> inner) {
        super(new TimeWindowedSerializer<>(inner.serializer()), new TimeWindowedDeserializer<>(inner.
deserializer()));
}
```

Overloading this constructor would allow us to pass in an explicit time window size without changing the existing constructor.

In WindowedSerdes, we add a new method to return a TimeWindowSerde with configurable window size:

```
static public <T> Serde<Windowed<T>> timeWindowedSerdeFrom(final Class<T> type, final long windowSize)
```

In `TimeWindowedSerde`, we will add an additional constructor (with an internal boolean field, *isChangelogTopic*, set to false by default in the deserializer), as well as a helper method, *forChangelog* to set the isChangelog flag. We introduce a new public method *forChangelog,* for users to explicitly set whether the input topic is a changelog topic or not so that windowed keys in a changelog topic could be deserialized properly. If users do not call *forChangelog* on a changelog input topic type, the windowed keys extracted will be invalid due to inconsistency in how they were serialized.

```
public TimeWindowedSerde(final Serde<T> inner, final long windowSize) {
        super(new TimeWindowedSerializer<>(inner.serializer()), new TimeWindowedDeserializer<>(inner.
deserializer(), windowSize));
}

// Helper method for advanced users who want to read from a changelog windowed topic
TimeWindowedSerde forChangelog(final boolean);
```

For example usage, we will allow users to pass in the serdes in `Consumed and` only require users to pass in the inner serde inside `consumed` parameter, and a library can wrap it with the TimeWindowedSerde and the window size, and call a method, forChangelog() explicitly, if the input is a changelog topic:

```
// Wrap the user provided serde with the window serde, with the window size, and set the changelog input topic
type.
TimeWindowedSerde windowedSerde = WindowedSerdes.timeWindowedSerdeFrom(consumed.keySerde, timeWindow.size()).
forChangelog(true);

final ConsumedInternal<K, V> consumedInternal =
      new ConsumedInternal<>(Consumed.with(windowedSerde, consumed.valueSerde()));
```

One remaining step is to let the TimeWindowedDeserializer be aware of the input topic type by creating a new constructor, and change the deserialize method to be aware of the changelog topic boolean flag.

```
public TimeWindowedDeserializer(final Deserializer<T> inner, final long windowSize, boolean isChangelogTopic) {
      this.inner = inner;
      this.windowSize = windowSize;
      this.isChangelogTopic = isChangelogTopic;
}

@Override
public Windowed<T> deserialize(final String topic, final byte[] data) {
        ...
    // toStoreKeyBinary was used to serialize the data.
    if (this.isChangelogTopic)
        return WindowKeySchema.fromStoreKey(data, windowSize, inner, topic);
        ...
}
```

Compatibility, Deprecation, and Migration Plan

- This KIP will not change the existing TimeWindowed serde, but rather just extend it. This change should be backwards compatible.

Rejected Alternatives

We could introduce a new serde, TimeWindowedChangelogSerde to explicitly serialize and deserialize changelog input topic.

This would require an additional serde that does a very similar job to TimeWindowedSerde. The only problem is the way we deserialize a different input source type (changelog topic), so instead it would be cleaner to just overload the TimeWindowedSerde to have an additional parameter. There is also inconsistency in how we serialize a changelog topic key and how we deserialize keys in the TimeWindowedSerde. Introducing a new changelog serde does not fix the inherit issue in the TimeWindowedSerde. We do not want users to be aware of the implementation details of the two serdes.