

KIP-466: Add support for List<T> serialization and deserialization

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

Status

Current state: *Adopted*

Discussion thread: [Link](#)

JIRA:

 Unable to render Jira issues macro, execution error.

Motivation

I believe there are many use cases where List Serde could be useful:

- <https://stackoverflow.com/questions/41427174/aggregate-java-objects-in-a-list-with-kafka-streams-dsl-windows>
- <https://stackoverflow.com/questions/46365884/issue-with-arraylist-serde-in-kafka-streams-api>

For instance, aggregate grouped (by key) values together in a list to do other subsequent operations on the collection.

Public Interfaces

- New class `org.apache.kafka.common.serialization.ListSerializer<Inner>` implements `Serializer<List<Inner>>` interface
- New class `org.apache.kafka.common.serialization.ListDeserializer<Inner>` implements `Deserializer<List<Inner>>` interface
- New static subclass class `ListSerde<Inner>` extends `WrapperSerde<List<Inner>>` in `org.apache.kafka.common.serialization.Serdes`
- New method `public static <L extends List, Inner> Serde<List<Inner>> ListSerde(Class<L> listClass, Serde<Inner> innerSerde)` in `org.apache.kafka.common.serialization.Serdes` class

Proposed Changes

This KIP proposes adding new `ListSerializer` and `ListDeserializer` classes as well as support for the new `ListSerde` nested class inside the `Serdes` class. This will allow using `Serde<List<Inner>>` directly from Consumers, Producers and Streams.

`Serde<List<Inner>>` serialization and deserialization will be done through repeatedly calling a serializer/deserializer for each entry provided by passed generic `Inner`'s `serde`. For example, if you want to create List of Strings `serde`, then serializer/deserializer of `Serdes.StringSerde` will be used to serialize /deserialize each entry in `List<String>`.

Proposed Configurations

List `serde` is an unusual type of `serde` because we need to consider two things here: the implementation of `List` interface (i.e. `ArrayList`, `LinkedList`, etc) as **well as** its enclosed elements' type.

First, we need to specify that we are going to use a list `serde`:

```
default.key/value.serde = org.apache.kafka.common.serialization.Serdes$ListSerde
```

Then, we need to introduce two brand new configurations and here I'm proposing these four extra properties:

```
CommonClientConfigs.class: DEFAULT_LIST_KEY/VALUE_SERDE_TYPE_CLASS = "default.list.key.serde.type"

Ex. default.list.key/value.serde.type = java.util.ArrayList
```

```
CommonClientConfigs.class: DEFAULT_LIST_KEY/VALUE_SERDE_INNER_CLASS = "default.list.key.serde.inner"

Ex. default.list.key/value.serde.inner = org.apache.kafka.common.serialization.Serdes$IntegerSerde
```

P.S. Properties **default.list.key/value.*** will be ignored as long as **default.key/value.serde** is not set to **org.apache.kafka.common.serialization.Serdes\$ListSerde**

Serialization Strategy

For the performance purposes the following serialization strategies were put in place:

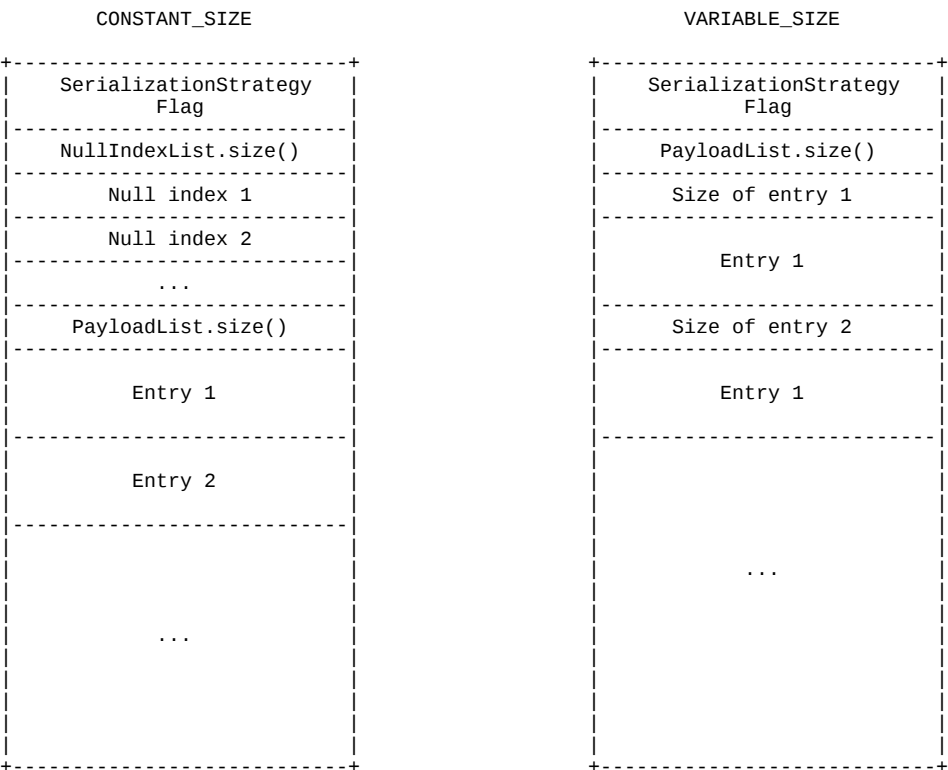
```
enum SerializationStrategy {
    CONSTANT_SIZE,
    VARIABLE_SIZE;
}
```

Depending on the type of an inner serde (a list's element type) the serialization will be performed in the following ways:

- 1. For `SerializationStrategy.CONSTANT_SIZE`, if an inner serde has one of the following serializers (`ShortSerializer.class`, `IntegerSerializer.class`, `FloatSerializer.class`, `LongSerializer.class`, `DoubleSerializer.class`, `UUIDSerializer.class`), then the final payload **will not** contain each **element's size encoded** since sizes of presented types are static (2 bytes, 4 bytes, 8 bytes, etc.)
- 2. For `SerializationStrategy.VARIABLE_SIZE`, if the inner serde doesn't have one of the serializers listed above, then a size of each element will be encoded in the final payload (see below)

Additionally, there are two different ways of serializing NULL values within the payload:

- 1. For `SerializationStrategy.CONSTANT_SIZE`, the list serializer will generate a null index list that contains indexes of all null entries within the payload
- 2. For `SerializationStrategy.VARIABLE_SIZE`, the list serializer instead will write `Serdes.ListSerde.NULL_ENTRY_VALUE` (-1 by default) for the size of a null entry



Compatibility, Deprecation, and Migration Plan

Does not apply

Rejected Alternatives

Not known