

 [+1\) 647-467-4396 \(tel:16474674396\)](tel:16474674396) [hello@knoldus.com \(mailto:hello@knoldus.com\)](mailto:hello@knoldus.com)[\(https://blog.knoldus.com/\)](https://blog.knoldus.com/)

Kafka – Sending Object as a message

 [January 30, 2017 \(https://blog.knoldus.com/kafka-sending-object-as-a-message/\)](https://blog.knoldus.com/kafka-sending-object-as-a-message/)  [Prabhat](#)[Kashyap \(https://blog.knoldus.com/author/prabhatkashyap92/\)](https://blog.knoldus.com/author/prabhatkashyap92/) [Apache Kafka \(https://blog.knoldus.com/category/tech-blogs/big-data/apache-kafka/\)](https://blog.knoldus.com/category/tech-blogs/big-data/apache-kafka/), [Java \(https://blog.knoldus.com/category/tech-blogs/java/\)](https://blog.knoldus.com/category/tech-blogs/java/) [3 Comments \(https://blog.knoldus.com/kafka-sending-object-as-a-message/#comments\)](https://blog.knoldus.com/kafka-sending-object-as-a-message/#comments)



Kafka lets us publish and subscribe to streams of records and the records can be of any type, it can be JSON, String, POJO, etc. Kafka gives user the ability to create our own serializer and deserializer so that we can transmit different data type using it. In this blog I will demonstrate how to create a custom serializer and deserializer but first let's understand what is serialization and why to serialize?

Serialization and Deserialization

Serialization is the process of converting an object into a stream of bytes and that bytes are used for transmission. Kafka stores and transmit these bytes of array in its queue.

Deserialization as the name suggest does the opposite of serialization where we convert bytes of array into the desired data type. Kafka provides serializer and deserializer for few data types ***String, Long, Double, Integer, Bytes etc.***

Check all pre-build (de) serializers :

<https://kafka.apache.org/0100/javadoc/org/apache/kafka/common/serialization/package-summary.html>

(<https://kafka.apache.org/0100/javadoc/org/apache/kafka/common/serialization/package-summary.html>).



Now I hope you understand what is **serialization** and why we serialize any object, so let's begin with its implementation.

Implementation

To create serializer class we need to implement

org.apache.kafka.common.serialization.Serializer interface and similarly to create deserializer class we need to implement

org.apache.kafka.common.serialization.Deserializer interface.

Both serializer and deserializer interfaces consist of three methods:

- **configure** : This method called at startup with configuration.
- **serialize/deserialize** : This method is used for serialization and deserialization.
- **close** : This method called when Kafka session is to be closed.

Serializer Interface

```
public interface Serializer extends Closeable {  
    void configure(Map<String, ?> var1, boolean var2);  
  
    byte[] serialize(String var1, T var2);  
  
    void close();  
}
```

Deserializer Interface



```
public interface Deserializer extends Closeable {  
    void configure(Map<String, ?> var1, boolean var2);  
  
    T deserialize(String var1, byte[] var2);  
  
    void close();  
}
```

Let's start with an example:

Dependencies I've used:

- Kafka – 0.10.1.1
- FasterXML jackson – 2.8.6

User.java



```
public class User {  
  
    private String name;  
    private int age;  
  
    public User() {  
    }  
  
    public User(String name, int age) {  
        this.name = name;  
        this.age = age;  
    }  
  
    public String getName() {  
        return this.name;  
    }  
  
    public int getAge() {  
        return this.age;  
    }  
  
    @Override public String toString() {  
        return "User(" + name + ", " + age + ")";  
    }  
}
```

UserSerializer.java



```
public class UserSerializer implements Serializer {

    @Override public void configure(Map<String, ?> map, boolean b) {

    }

    @Override public byte[] serialize(String arg0, User arg1) {
        byte[] retVal = null;
        ObjectMapper objectMapper = new ObjectMapper();
        try {
            retVal = objectMapper.writeValueAsString(arg1).getBytes();
        } catch (Exception e) {
            e.printStackTrace();
        }
        return retVal;
    }

    @Override public void close() {

    }

}
```

UserDeserializer.java



```
public class UserDeserializer implements Deserializer {

    @Override public void close() {

    }

    @Override public void configure(Map<String, ?> arg0, boolean arg1) {

    }

    @Override
    public User deserialize(String arg0, byte[] arg1) {
        ObjectMapper mapper = new ObjectMapper();
        User user = null;
        try {
            user = mapper.readValue(arg1, User.class);
        } catch (Exception e) {

            e.printStackTrace();
        }
        return user;
    }

}
```

Now what's left is to use these **serializer** and **deserializer**.

To use above serializer we need to register this property:



```
props.put("value.serializer", "com.knoldus.serializers.UserSerializer")
```

Using this property, producer will be :

```
try (Producer<String, User> producer = new KafkaProducer<>(props)) {  
    producer.send(new ProducerRecord<String, User>("MyTopic", user));  
    System.out.println("Message " + user.toString() + " sent !!");  
} catch (Exception e) {  
    e.printStackTrace();  
}
```

Similarly for deserializer we need to register this property:

```
props.put("value.deserializer", "com.knoldus.deserializer.UserDeseriali
```

And consumer will be :




```
try (KafkaConsumer<String, User> consumer = new KafkaConsumer<>(props))
    consumer.subscribe(Collections.singletonList(topic));
    while (true) {
        ConsumerRecords<String, User> messages = consumer.poll(100);
        for (ConsumerRecord<String, User> message : messages) {
            System.out.println("Message received " + message.value().toSt
        }
    }
} catch (Exception e) {
    e.printStackTrace();
}
```

References:

<https://kafka.apache.org/>(<https://kafka.apache.org/>).



(<http://www.knoldus.com/connect/contactus.knol>).

Like 0

Share the Knol:



Share

Share 0

Tweet



WhatsApp (<https://blog.knoldus.com/kafka-sending-object-as-a-message/?share=jetpack-whatsapp&nb=1>)



Telegram (<https://blog.knoldus.com/kafka-sending-object-as-a-message/?share=telegram&nb=1>)



More



Written by Prabhat Kashyap

(<https://blog.knoldus.com/author/prabhatkashyap92/>)

Prabhat is a Sr. Software Consultant with more than 2 years of experience in C, C++, Java, MySQL, and Scala. His interests are in Cyber Security, Web Development, and New technologies. Prabhat developed software and website on different platforms which include VB6, VB.NET, ASP.NET, PHP, Wordpress, OpenCart, SMF, VBulletin, HTML5, MyBB. Prabhat is focused and result oriented, self-motivated and team-oriented and effective team player.

3 thoughts on “Kafka – Sending Object as a message”





Himani Arora (<http://himaniarora1.wordpress.com>) says:

📅 January 30, 2017 at 3:49 PM (<https://blog.knoldus.com/kafka-sending-object-as-a-message/#comment-30895>)

Reblogged this on [himaniarora1 \(https://himaniarora1.wordpress.com/2017/01/30/kafka-sending-object-as-a-message/\)](https://himaniarora1.wordpress.com/2017/01/30/kafka-sending-object-as-a-message/).

★ Loading...



Rodney Barbati says:

📅 May 14, 2018 at 10:14 PM (<https://blog.knoldus.com/kafka-sending-object-as-a-message/#comment-36738>)

If the only way to serialize / deserialize objects is to have a custom serializer/deserializer for each class, then this is not a solution for large scale applications. I think that if every class carries its own name as a data member within itself, then a generic serializer/deserializer should be possible – first reading the name of the class from the byte stream and using it to cast the result to the correct type.

★ Loading...



Maxim Neaga (<https://plus.google.com/+MaximNeaga>) says:

📅 December 1, 2018 at 2:38 AM (<https://blog.knoldus.com/kafka-sending-object-as-a-message/#comment-38633>)



Or serialize objects into JSON with Jackson.

★ Loading...

Comments are closed.

Knoldus is the world's largest pure-play Scala and Spark company. We modernize enterprise through cutting-edge digital engineering by leveraging Scala, Functional Java and Spark ecosystem. Our mission is to provide reactive and streaming fast data solutions that are message-driven, elastic, resilient, and responsive.

Follow Us

f [_ \(https://www.facebook.com/KnoldusSoftware/\)](https://www.facebook.com/KnoldusSoftware/)

t [_ \(https://twitter.com/Knolspeak\)](https://twitter.com/Knolspeak)

in [_ \(https://in.linkedin.com/company/knoldus\)](https://in.linkedin.com/company/knoldus)

About Knoldus



[Company \(/about/historyideology.knol\)](/about/historyideology.knol)

[Events \(/about/events-conference.knol\)](/about/events-conference.knol)

[Why Knoldus \(/connect/why-knoldus.knol\)](/connect/why-knoldus.knol)

[Careers \(/connect/careers.knol\)](/connect/careers.knol)

[Services \(/services/overview.knol\)](/services/overview.knol)

[Case Studies \(/work/case-studies.knol\)](/work/case-studies.knol)

[Products \(http://getcodesquad.com/\)](http://getcodesquad.com/)

[Process \(/about/process.knol\)](/about/process.knol)

[Resources \(/learn/resources.knol\)](/learn/resources.knol)

[Terms \(/terms.knol\)](/terms.knol)

[News \(/news.knol\)](/news.knol)

[Site Map \(/sitemap/sitemap.knol\)](/sitemap/sitemap.knol)

[Contact us \(/connect/contact-us.knol\)](/connect/contact-us.knol)

[Engagement Model \(/connect/engagement-models.knol\)](/connect/engagement-models.knol)

Recent Posts

[Flinkathon: What makes Flink better than Kafka Streams?](#)

16-4-2019

[\(https://blog.knoldus.com/flinkathon-what-makes-flink-better-than-kafka-streams/\)](https://blog.knoldus.com/flinkathon-what-makes-flink-better-than-kafka-streams/)

[Knolx: Structured Streaming in Spark](#)

15-4-2019

[\(https://blog.knoldus.com/knolx-structured-streaming-in-spark/\)](https://blog.knoldus.com/knolx-structured-streaming-in-spark/)

[Scala: Type Bounds](#)

15-4-2019

[\(https://blog.knoldus.com/scala-type-bounds/\)](https://blog.knoldus.com/scala-type-bounds/)

Subscribe to our newsletter

Contact us

+(1) 647-467-4396



hello@knoldus.com

PARTNERS

<https://www.lightbend.com/partners/system-integrators>

<https://databricks.com/spark/certification/certified-system-integrators>

<https://www-356.ibm.com/partnerworld/wps/servlet/ContentHandler/partnerworld-home>

<https://www.datastax.com/datastax-partner-program>

<http://www.confluent.io/partners/>

Copyright 2017-19 © Knoldus

