[Apache Kafka](https://kafka.apache.org/" \t "_blank)applications run in a distributed manner across multiple containers or machines. It is necessary to handle error while producing data in Kafka producer to prevent data loss.

When the producer sends messages to a broker, the broker can return either a success or an error code.

Those error codes belong to two categories.

* **Retriable errors**. Errors that can be resolved after retrying. For example, if the broker returns the exception **NotEnoughReplicasException**, the producer can try sending the message again - maybe replica brokers will come back online, and the second attempt will succeed.
* **NonRetriable error.**Errors that won’t be resolved. For example, if the broker returns an **INVALID\_CONFIG** exception, for an example **Topic not found**, trying the same producer request again will not change the outcome of the request.

Producer failure Handling is described below for the above 2 different cases.

* Retry Policy
* Callback

**Retry Policy**

* For Retriable errors Kafka has inbuilt producer properties to enable retry behaviour

**retries**

The retries setting determines how many times the producer will attempt to send a message before marking it as failed. The default values are:

* 0 for Kafka <= 2.0
* MAX\_INT, i.e., 2147483647 for Kafka >= 2.1

Users should generally prefer to leave this config unset and instead use delivery.timeout.ms to control retry behaviour.

**delivery.timeout.ms**

If retries > 0, for example, retries = 2147483647, the producer won’t try the request forever as it’s bounded by a timeout. If delivery.timeout.ms = 1200000 ms then timeout for the producer is 2 min.

**retry.backoff.ms**

By default, the producer will wait 100ms between retries, but you can control this using the retry.backoff.ms parameter.

**idempotent**

if idempotent is set as **true** then the producer makes sure that exactly only one message written to the Kafka.

**max.in.flight.requests.per.connection**

Allowing retries without setting max.in.flight.requests.per.connection to 1 will potentially change the ordering of the messages. By limiting the number of in-flight requests to 1 (default being 5), i.e., max.in.flight.requests.per.connection = 1, Kafka will preserve the message order (key-based) in the event.

But we can resolve this **message disorder issue** with the help of **‘Idempotent Producer’** feature which is available since Kafka’s version 0.11.

if we enable idempotence enable.idempotence.config=true, then it is required for max.in.flight.requests.per.connection to be less than or equal to 5

## CallBack

Even after retrying if any data not sent to topic or if any error occurred due to non-retriable error eg: **Topic not found**.

Kafka producer client provides a way to add **producer callback**, by which we can capture response. Based on the response we can customize our own action.

**Producer timeout**

Retries are not infinite and are constrained by a timeout mechanism. With the advent of Kafka 2.1, a new, user-friendly timeout feature, known as the delivery timeout millisecond, was set by default to 120,000 milliseconds, or two minutes. This delivery timeout encompasses all prior timeouts, offering a straightforward approach. **From the moment a message is sent until its acknowledgment sent by Kafka Broker, the entire process is governed by this 120,000-millisecond delivery timeout.**

A diagram of a delivery process

Description automatically generated

Therefore, in analysing this mechanism, it’s essential to focus on the overall delivery timeout rather than the individual timeouts that preceded it. If a message will not be acknowledged within this timeframe, then it will be considered a failure.