# KIP-582: Add a "continue" option for Kafka Connect error handling



## Status

**Current state**: Under Discussion

**Discussion thread**: here

**JIRA**: here

**PR:** here (WIP)

## Motivation

Currently there are two error handling options in Kafka Connect, “none” and “all”. Option “none” will config the connector to fail fast, and option “all” will ignore broken records.

If users want to store their broken records, they have to config a broken record queue, which is too much work for them in some cases. The complexity comes from maintaining additional topics and connectors, which requires extra time and money. I can imagine a case where a user has 3 topics consumed by S3, HDFS and JDBC respectively. The user has to maintain 3 more connectors to consume three DLQs, in order to put broken records to the place they should go. This new option will give users a choice to only maintain half of their connectors, yet having broken records stored in each destination system.

Some sink connectors have the ability to deal with broken records, for example, a JDBC sink connector can store the broken raw bytes into a separate table, a S3 connector can store that in a zipped file.

Therefore, it would be idea if Kafka Connect provides an additional option that sends the broken raw bytes to SinkTask directly.

## Public Interfaces

In Kafka Connect, the configuration `errors.tolerance` will have a third option "continue" besides "none" and "all".

SinkTask will be added a new function to handle broken records

```
public void putBrokenRecord(Collection<SinkRecord> var1) {
  throw UnsupportedOperationException("Broken record handling is not supported by this connector"); // This is the default implementation.
}
```

Connectors can overwrite this function to handle broken records.

## Proposed Changes

Add a third option to error handling, which should behave like “continue” when error occurs at Converter or SMT. The infrastructure should send the broken byte message directly to SinkTask.

Schema of the broken bytes should be `Optional Byte Array`, and the value and the key would be raw bytes that we received from Kafka.

SinkTask is then responsible for handling the unparsed bytes input.

The benefits of having this additional option are:

- Being user friendly. Connectors can handle broken record and hide that from clients.
- Providing more flexibility to SinkTask in terms of broken record handling.

## Compatibility, Deprecation, and Migration Plan

There is no compatibility issue. Behavior of Kafka Connect does not change unless user explicitly specify the error handling method to be "continue".

# Rejected Alternatives

As suggested by Chris in the email discussion, we have a rejected alternative plan to this. I quoted the plan below:

> Configure your connector to send malformed records to a dead letter queue topic, and configure a separate connector to consume from that dead letter queue topic, use the ByteArrayConverter to deserialize records, and send those records to the destination sink.

Compared with this alternative, the point of this proposal is to save the effort to maintain twice as many connectors as necessary.