KIP-334 - Include partitions in exceptions raised during consumer record deserialization/validation

Status

Current state: *Accepted*

Discussion thread: here *[Change the link from the KIP proposal email archive to your own email thread]*

Vote thread: here

JIRA: *KAFKA-5682*

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

Motivation

Whenever consumers encounter an exception caused by record deserialization or record invalidity (e.g corrupt record), a `SerializationException` or `KafkaException` is raised to the user, with a string message similar to this: "Error deserializing key/value for partition test-0 at offset 10. If needed, please seek past the record to continue consumption."

The problem is that while the topic, partition and offset are reported in the message, they are not directly available in the exception - leaving the user unable to seek past the record, since he does not know which partition is the cause. (unless he parses the string message)

Public Interfaces

```
package org.apache.kafka.common.errors;

import org.apache.kafka.common.TopicPartition;

/**
 *  Any exception encountered on an invalid record (deserialization error, corrupt record, etc...)
 */
public class RecordDeserializationException extends SerializationException {

    private static final long serialVersionUID = 1L;
    private TopicPartition partition;
    private long offset;

    public RecordDeserializationException(TopicPartition partition, long offset, String message) {
        this(partition, offset, message, null);
    }

    public RecordDeserializationException(TopicPartition partition, long offset, String message, Throwable
cause) {
        super(message, cause);
        this.partition = partition;
        this.offset = offset;
    }

    public TopicPartition partition() {
        return partition;
    }

    public long offset() {
        return offset;
    }

    /* avoid the expensive and useless stack trace for deserialization exceptions */
    @Override
    public Throwable fillInStackTrace() {
        return this;
    }
}
```

Proposed Changes

Introduce one new public exception - \`RecordDeserializationException\` which extends \`SerializationException\` (and indirectly extends \`KafkaException\`). This new exceptions will hold two additional parameters - \`TopicPartition partition\` and \`long offset\`, allowing the user to catch it and access the metadata about the record which caused the exception.

This exception will be raised instead of \`SerializationException\` and \`KafkaException\` for invalid deserialization and invalid record handling logic, in the \`Fetcher.java\` class, essentially being propagated to the Consumer API and allowing the user to handle them by accessing the partition/offset to seek past the faulty record.

Example

Users are expected to seek one offset past the faulty one.

```
while (true) {
    try {
        consumer.poll(Duration.ofMillis(50));
    } catch (RecordDeserializationException e) {
        consumer.seek(e.partition(), e.offset() + 1);
    }
}
```

Compatibility, Deprecation, and Migration Plan

This is a fully backwards-compatible change. Since the new exceptions extend the original exception classes, existing logic which handles the original exceptions will continue handling the new ones.

Rejected Alternatives

- *Add `TopicPartition partition` and `long offset` attributes to the existing `SerializationException` and `KafkaException`*
    - *They will still be backwards-compatible, but might easily result in `NullPointerException`, since not every use case (especially for `KafkaException`) has the appropriate offset.*
    - *Users will need to check for null before using the new attributes, which is very error-prone*
- *Create two new exception classes `RecordDeserializationException` and `InoperativeRecordException`*
    - *Kafka's exception hierarchy is currently not ideal and it's best not to expose too much public exceptions as this limits what we can change in the internals.*
    - *The current proposal is easier to implement from a user's perspective*
- *Create new `DeserializationException` which does not extend `SerializationException`.*
    - *Will be backwards-incompatible*

Future Work

It would be useful to seek past such failed messages automatically. It is worth investigating the idea of adding a configurable default callback or config option to enable users to do that without having to manually implement error handling code.