# KIP-216: IQ should throw different exceptions for different errors



## Status

**Current state**: *Under Discussion*

**Discussion thread**: *here*

**JIRA**: KAFKA-5876 - Getting issue details... STATUS

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

## Motivation

Currently, IQ throws ***InvalidStateStoreException*** for any types of error, that means a user cannot handle different types of error.

Because of that, we should throw different exceptions for each type.

# Proposed Changes

To distinguish different types of error, we need to handle all ***InvalidStateStoreException***s better during these public methods invoked. The main change is to introduce new exceptions that extend from *InvalidStateStoreException*. *InvalidStateStoreException* is not thrown at all anymore, but only new sub-classes.

```
# Two category exceptions
public class RetryableStateStoreException extends InvalidStateStoreException
public class FatalStateStoreException extends InvalidStateStoreException


# Retryable exceptions
public class StreamThreadNotStartedException extends RetryableStateStoreException
public class StreamThreadRebalancingException extends RetryableStateStoreException
public class StateStoreMigratedException extends RetryableStateStoreException


# Fatal exceptions
public class StreamThreadNotRunningException extends FatalStateStoreException
```

Various state store exceptions can classify into two category exceptions: **RetryableStateStoreException** and **FatalStateStoreException**. The user can use the two exceptions if they only need to distinguish whether it can retry.

- Retryable exceptions
  - **StreamThreadNotStartedException**: will be thrown when streams thread state is CREATED, the user can retry until to RUNNING.
  - **StreamThreadRebalancingException**: will be thrown when stream thread is not running and stream state is REBALANCING, the user just retry and wait until rebalance finished (RUNNING).
  - **StateStoreMigratedException**: will be thrown when state store already closed and stream state is REBALANCING.
- Fatal exceptions
  - **KafkaStreamsNotRunningException**: will be thrown when stream thread is not running and stream state is PENDING_SHUTDOWN / NOT_RUNNING / ERROR. The user cannot retry when this exception is thrown.
  - **StateStoreNotAvailableException**: will be thrown when state store closed and stream thread is PENDING_SHUTDOWN / NOT_RUNNING / ERROR. The user cannot retry when this exception is thrown.
  - **UnknownStateStoreException**: will be thrown when passing an unknown state store.

The following is the public methods that users will call to get state store instance:

- *KafkaStreams*
  - store(storeName, queryableStoreType)

Throw exceptions: **StreamThreadNotStartedException**, **StreamThreadRebalancingException**, **KafkaStreamsNotRunningException, UnknownStateStoreException**

The following is the public methods that users will call to get store values:

- *interface ReadOnlyKeyValueStore(class CompositeReadOnlyKeyValueStore)*
  - get(key)
  - range(from, to)
  - all()
  - approximateNumEntries()
- *interface ReadOnlySessionStore(class CompositeReadOnlySessionStore)*
  - fetch(key)
  - fetch(from, to)
- *interface ReadOnlyWindowStore(class CompositeReadOnlyWindowStore)*
  - fetch(key, time)
  - fetch(key, from, to)
  - fetch(from, to, fromTime, toTime)
  - all()
  - fetchAll(from, to)
  - @Deprecated fetch(key, timeFrom, timeTo)
  - @Deprecated fetch(from, to, timeFrom, timeTo)
  - @Deprecated fetchAll(timeFrom, timeTo)
- *interface KeyValueIterator(class DelegatingPeekingKeyValueIterator)*
  - next()
  - hasNext()
  - peekNextKey()

All the above methods could be throw following exceptions:

***StreamThreadRebalancingException, StateStoreMigratedException, KafkaStreamsNotRunningException, StateStoreNotAvailableExceptio***

# Compatibility, Deprecation, and Migration Plan

- All new exceptions extend from InvalidStateStoreException, this change will be fully backward compatible.

# Rejected Alternatives

None.