/\*\*

\* Copyright 2021 Twitter, Inc.

\* SPDX-License-Identifier: Apache-2.0

\*/

package com.twitter.unified\_user\_actions.kafka.serde.internal

import com.google.common.util.concurrent.RateLimiter

import com.twitter.finagle.stats.Counter

import com.twitter.finagle.stats.NullStatsReceiver

import java.util

import com.twitter.scrooge.CompactThriftSerializer

import com.twitter.scrooge.ThriftStruct

import com.twitter.scrooge.ThriftStructCodec

import com.twitter.scrooge.ThriftStructSerializer

import org.apache.kafka.common.serialization.Deserializer

import org.apache.kafka.common.serialization.Serde

import org.apache.kafka.common.serialization.Serializer

import com.twitter.util.logging.Logging

import org.apache.thrift.protocol.TBinaryProtocol

abstract class AbstractScroogeSerDe[T <: ThriftStruct: Manifest](nullCounter: Counter)

extends Serde[T]

with Logging {

private val rateLimiter = RateLimiter.create(1.0) // at most 1 log message per second

private def rateLimitedLogError(e: Exception): Unit =

if (rateLimiter.tryAcquire()) {

logger.error(e.getMessage, e)

}

private[kafka] val thriftStructSerializer: ThriftStructSerializer[T] = {

val clazz = manifest.runtimeClass.asInstanceOf[Class[T]]

val codec = ThriftStructCodec.forStructClass(clazz)

constructThriftStructSerializer(clazz, codec)

}

private val \_deserializer = new Deserializer[T] {

override def configure(configs: util.Map[String, \_], isKey: Boolean): Unit = {}

override def close(): Unit = {}

override def deserialize(topic: String, data: Array[Byte]): T = {

if (data == null) {

null.asInstanceOf[T]

} else {

try {

thriftStructSerializer.fromBytes(data)

} catch {

case e: Exception =>

nullCounter.incr()

rateLimitedLogError(e)

null.asInstanceOf[T]

}

}

}

}

private val \_serializer = new Serializer[T] {

override def configure(configs: util.Map[String, \_], isKey: Boolean): Unit = {}

override def serialize(topic: String, data: T): Array[Byte] = {

if (data == null) {

null

} else {

thriftStructSerializer.toBytes(data)

}

}

override def close(): Unit = {}

}

/\* Public \*/

override def configure(configs: util.Map[String, \_], isKey: Boolean): Unit = {}

override def close(): Unit = {}

override def deserializer: Deserializer[T] = {

\_deserializer

}

override def serializer: Serializer[T] = {

\_serializer

}

/\*\*

\* Subclasses should implement this method and provide a concrete ThriftStructSerializer

\*/

protected[this] def constructThriftStructSerializer(

thriftStructClass: Class[T],

thriftStructCodec: ThriftStructCodec[T]

): ThriftStructSerializer[T]

}

class ThriftSerDe[T <: ThriftStruct: Manifest](nullCounter: Counter = NullStatsReceiver.NullCounter)

extends AbstractScroogeSerDe[T](nullCounter = nullCounter) {

protected[this] override def constructThriftStructSerializer(

thriftStructClass: Class[T],

thriftStructCodec: ThriftStructCodec[T]

): ThriftStructSerializer[T] = {

new ThriftStructSerializer[T] {

override val protocolFactory = new TBinaryProtocol.Factory

override def codec: ThriftStructCodec[T] = thriftStructCodec

}

}

}

class CompactThriftSerDe[T <: ThriftStruct: Manifest](

nullCounter: Counter = NullStatsReceiver.NullCounter)

extends AbstractScroogeSerDe[T](nullCounter = nullCounter) {

override protected[this] def constructThriftStructSerializer(

thriftStructClass: Class[T],

thriftStructCodec: ThriftStructCodec[T]

): ThriftStructSerializer[T] = {

new CompactThriftSerializer[T] {

override def codec: ThriftStructCodec[T] = thriftStructCodec

}

}

}