package com.twitter.unified\_user\_actions.adapter.uua\_aggregates

import com.twitter.finagle.stats.NullStatsReceiver

import com.twitter.finagle.stats.StatsReceiver

import com.twitter.unified\_user\_actions.adapter.AbstractAdapter

import com.twitter.unified\_user\_actions.thriftscala.\_

/\*\*

\* The main purpose of the rekey adapter and the rekey service is to not break the existing

\* customers with the existing Unkeyed and also making the value as a super light-weight schema.

\* After we rekey from Unkeyed to Long (tweetId), downstream KafkaStreams can directly consume

\* without repartitioning.

\*/

class RekeyUuaAdapter extends AbstractAdapter[UnifiedUserAction, Long, KeyedUuaTweet] {

import RekeyUuaAdapter.\_

override def adaptOneToKeyedMany(

input: UnifiedUserAction,

statsReceiver: StatsReceiver = NullStatsReceiver

): Seq[(Long, KeyedUuaTweet)] =

adaptEvent(input).map { e => (e.tweetId, e) }

}

object RekeyUuaAdapter {

def adaptEvent(e: UnifiedUserAction): Seq[KeyedUuaTweet] =

Option(e).flatMap { e =>

e.actionType match {

case ActionType.ClientTweetRenderImpression =>

ClientTweetRenderImpressionUua.getRekeyedUUA(e)

case \_ => None

}

}.toSeq

}